Turbo codes et estimation paramétrique pour les communications à haut débit

Christophe Vanstraceele

N° ENSC-2005 n°34

THESE DE DOCTORAT
DE L'ECOLE NORMALE SUPERIEURE DE CACHAN

Présentée par

Christophe VANSTRACEELE

pour obtenir le grade de

DOCTEUR DE L'ECOLE NORMALE SUPERIEURE DE CACHAN

Domaine :
Electrotechnique-Electronique-Automatique
(Spécialité Traitement du Signal)

Sujet de la thèse :

Turbo Codes et estimation paramétrique pour les communications à haut débit

Thèse présentée et soutenue à Cachan le 26 janvier 2005 devant le jury composé de :

Jean-François HELARD	Président
Ramesh PYNDIAH	Rapporteur
Luc VANDENDORPE	Rapporteur
Jean-Marc BROSSIER	Examinateur
Pascal LARZABAL	Directeur de thèse
Benoit GELLER	Encadrant
Jean-Pierre BARBOT	Encadrant

Laboratoire SATIE
ENS CACHAN/CNRS/UMR 8029
61, avenue du Président Wilson, 94235 CACHAN CEDEX (France)

Remerciements

Mes premiers remerciements vont à Pascal Larzabal, qui m'a accueilli au sein de l'exceptionnelle équipe Traitement de Signal du SATIE.

Je tiens à exprimer ma profonde gratitude à Benoit Geller, qui a su me faire bénéficier de son expérience par ses précieux conseils durant ces trois années. Je le remercie pour la confiance qu'il m'a témoignée en m'accordant une grande autonomie et pour son humour qui participe à l'ambiance détendue de cette équipe.

Je remercie également Jean-Pierre Barbot pour sa bonne humeur communicative et ses suggestions visant à améliorer ce mémoire.

Je tiens à exprimer ma reconnaissance à Jean-Marc Brossier pour les discussions enrichissantes qui m'ont permis de faire avancer ce travail.

Merci également à Ramesh Pyndiah et Luc Vandendorpe pour m'avoir fait l'honneur d'être rapporteurs de ma thèse, et à Jean-François Hélard pour avoir présidé mon jury de thèse.

Merci à Gwénaël Robin pour avoir débugué mon ordinateur environ une fois par semaine.

Je remercie tous les rigolos de l'équipe, François, Marc, Angela, Ana et particulièrement Alex pour les longues nuits passées avec un pad dans une main et une pizza dans l'autre à refaire le monde sous l'effet de boissons exotiques.

Merci à la région de Franche-Comté pour ses saucisses, ses spécialités fromagères et son air revigorant durant mes week-ends sans lesquels je n'aurais pu abattre autant de travail.

Une pensée particulière pour mon épouse Géraldine, pour son soutien et pour avoir supporté l'éloignement durant ces trois longues années.

A ma meilleure copine et épouse,
Géraldine.

Table des matières

Chapitre 1

Introduction

Sommaire

1.1 Préambule

Ce sujet de thèse a été initié par le contrat européen MEDEA+ A106 INCA dirigé par ST Microélectronique, dans lequel était engagé le laboratoire Systèmes et Applications des Technologies de l'Information et de l'Energie (SATIE UMR 8029 CNRS/ENS Cachan), placé sous la co-tutelle de l'Ecole Normale Supérieure de Cachan et du CNRS. Cette étude a été menée au sein de l'équipe Transmission Large Bande de l'Information du pôle Traitement du Signal.

1.2 Motivations et cadre de l'étude

1.2.1 Contexte de l'étude

Le problème de la transmission d'information existe depuis que l'humanité existe. Que ce soit par la parole, par le courrier ou les livres, l'homme a toujours éprouvé le besoin de communiquer. Les exigences sans cesse grandissantes en matière de communication ont amené de nombreuses inventions au cours de l'histoire, telles que le morse et le téléphone.

La présentation de la théorie de l'information par Claude E. Shannon en 1948 [Sha48] a permis une nouvelle compréhension des systèmes de transmission. En définissant la

quantité maximale d'information pouvant être transmise sur un canal de transmission, C.E. Shannon a initié une évolution rapide des communications toujours en marche aujourd'hui. L'invention du transistor et des circuits intégrés a rendu possible les traitements de l'information nécessaires aux communications numériques. Ainsi, l'évolution est telle qu'aujourd'hui l'enregistrement de données analogiques et les communications analogiques sont en voie de disparition.

Les techniques numériques actuelles telles que le GPRS (General Packet Radio Service) pour les communications mobiles ou l'ADSL (Asymmetric Digital Subscriber Line) pour le réseau Internet ne constituent qu'une étape avant les prochaines générations, destinées elles aussi à devenir un jour obsolètes. Le marché des télécommunications étant de plus en plus soumis à la concurrence, l'avenir des opérateurs passe par la capacité à fournir à leurs abonnés de nouveaux services, de plus en plus performants. C'est dans ce contexte d'évolution permanente que prend place cette étude dont le but est d'introduire de nouvelles techniques utilisables dans les systèmes à venir.

1.2.2 Codage canal

Toute communication est perturbée par un bruit aléatoire présent sur le canal de transmission. La conséquence de ces perturbations est une dégradation de la transmission, qui se traduit par des erreurs à la réception. C'est dans le but de lutter contre ces erreurs que le codage canal a été conçu. Une redondance introduite de manière judicieuse dans le message par l'émetteur permet de corriger les erreurs au niveau du récepteur.

En 1948, C.E. Shannon avait prédit, grâce à sa théorie, les performances ultimes pouvant être atteintes par cette technique. Mais la théorie de l'information ne précisait pas comment atteindre ces performances. Dans un premier temps, deux familles de codes ont été inventées, les codes convolutifs et les codes blocs. Mais ces deux familles butaient contre une barrière de complexité sans arriver à s'approcher réellement de la limite prédite par C.E. Shannon.

En 1993, C. Berrou et A. Glavieux [BGT93] imaginèrent un schéma de codage et de décodage astucieux, les turbo codes. Basé sur l'utilisation de deux décodeurs convolutifs s'échangeant des informations, ce schéma atteignait cette limite, qui jusque là semblait impossible à atteindre. R. Pyndiah [Pyn98] a étendu cette technique aux codes blocs en inventant un schéma particulièrement simple à mettre en oeuvre. Dans cette étude, nous nous intéressons en particulier aux turbo codes blocs imaginés par R. Pyndiah. Notre but étant de trouver de nouvelles applications à ce schéma, nous exploitons les informations de fiabilité qu'il fournit afin d'affiner l'estimation d'autres paramètres nécessaires à une transmission.

1.2.3 Estimation paramétrique

Un système de communication numérique n'est pas perturbé uniquement par le bruit. Le canal de transmission présente d'autres défauts, comme par exemple sa fonction de transfert qui déforme le signal émis. L'emetteur utilisant une fréquence particulière pour moduler le signal à transmettre, il faut en réception être synchronisé avec l'horloge

utilisée par l'emetteur, sans quoi le signal reçu sera erroné. Tous ces paramètres, phase, fréquence, fonction de transfert ou instant d'échantillonnage, doivent être correctement estimés par des algorithmes afin d'être compensés et ainsi retrouver l'information émise. Mais dans des conditions difficiles, il est possible que les techniques actuelles soient insuffisantes. C'est dans cette optique que nous présentons des algorithmes d'estimation de phase, qui ont la particularité d'exploiter les informations disponibles à la sortie d'un turbo décodeur pour optimiser leurs performances.

1.3 Présentation du document

Outre ce premier chapitre d'introduction, ce document est constitué de cinq chapitres supplémentaires.

Le **deuxième chapitre** présente les principes des base des systèmes de communications numériques, dont la bonne compréhension est nécessaire pour aborder les chapitres suivants. Nous y introduisons des notions telles que la modulation QAM ou les modulations multiporteuses.

Le **troisième chapitre** se focalise sur le problème du codage canal. Après une description précise des codes cycliques, nous introduisons la notion de turbo décodeur bloc. Nous présentons dans ce chapitre une simplification particulièrement intéressante pour la mise en oeuvre des turbo décodeurs blocs.

Le **quatrième chapitre** constitue une étude concernant le système VDSL. Nous décrivons dans un premier temps les aspects importants de la norme. Ensuite, nous évaluons l'intérêt de l'utilisation des turbo codes blocs dans un tel contexte.

Le **cinquième chapitre** aborde le problème important de la synchronisation en phase. Après une brève présentation du problème, nous y présentons deux algorithmes de synchronisation exploitant les informations disponibles à la sortie d'un turbo décodeur afin d'optimiser la qualité de l'estimation.

Enfin, le **sixième chapitre** concluera sur l'état actuel de l'étude et dégagera certaines perspectives intéressantes.

1.4 Contributions

- Nous avons imaginé une mise en oeuvre plus légère en quantité de mémoire de l'algorithme de turbo décodage de R. Pyndiah. Il est ainsi possible de faciliter la fabrication d'un tel dispositif dans le but de l'intégrer dans un système réel. Ce dispositif a donné lieu au dépot d'un brevet [GBBV04a].

- Nous avons étudié l'interêt de l'utilisation des turbo codes blocs dans un système réel, le VDSL (Very high bit rate Digital Subscriber Line). La norme du VDSL première génération étant fixée, nous avons démontré qu'il serait intéressant d'utiliser ce schéma de codage dans la deuxième génération. Pour effectuer cette démonstration, nous avons participé au developpement d'une chaîne de simulation software entièrement écrite en C++, intégrant la norme VDSL, à savoir : codes correcteurs

d'erreur, et donc les turbo codes blocs, modulations QAM et DMT fixées par la norme, modèles de canaux filaires, bruits figurant dans la norme, ... Ce travail a été effectué en colaboration avec ST Microelectronics dans le cadre du projet européen MEDEA+ A106 (INCA). Les simulations ont été effectuées à l'aide d'un calculateur à 32 processeurs (Cluster de PC sous Linux), acheté dans le cadre de ce projet INCA par le laboratoire SATIE et dédié aux simulations des partenaires de ce projet.

- Un premier algorithme de synchronisation exploitant les informations fournies par le turbo décodeur pour estimer la phase sur un mot de code est présenté dans ce document. Il possède des caractéristiques intéressantes pouvant être utiles dans le cadre d'un système de communication haut débit. Un brevet concernant cet algorithme a été déposé [GBBV04c].

- Nous avons inventé un deuxième algorithme de synchronisation capable de manière conjointe d'estimer le gain du canal et ce sur chaque symbole transmis. Il exploite également les informations fournies par un turbo décodeur et peut poursuivre des évolutions particulièrement rapides de ces paramètres. Cet algorithme a également fait l'objet d'un dépot de brevet [GBBV04b].

1.5 Publications

Les travaux qui ont été effectués pendant cette thèse ont donné lieu à plusieurs brevets ainsi qu'à des publications :

- C. Vanstraceele, B. Geller, J. P. Barbot, J. M. Brossier, "Block Turbo Codes for Multicarrier Local Loop Transmission", Proc. VTC'2002 Fall, Vancouver, pp. 1775-1778, Vol 3, September, 2002.

- C. Vanstraceele, J. P. Barbot, B. Geller, J. M. Brossier, "BCH turbo codes for xDSL transmission", Proc.IEEE of InOWo'02, Hamburg, pp. 263-266, Vol 1, September, 2002.

- C. Vanstraceele, J. P. Barbot, L.N. Atallah, B. Geller, J. M. Brossier, "Estimation de l'erreur de phase d'une modulation QAM conjointement au décodage itératif d'un code bloc", Proc. GRETSI'2003 (19 th Symposium), Paris, pp. 363-366, Vol. 2, September, 2003.

- C. Vanstraceele, J. M. Brossier, J. P. Barbot, B. Geller, "A New Iterative Phase Tracking Scheme", Proc. ICICS'2003, Singapore, pp. 1-4, On CDroom 1C4.7, December, 2003.

- C. Vanstraceele, B. Geller, J. P. Barbot, J. M. Brossier, "Joint estimation of QAM carrier phase with block turbo decoding", Proc. IEEE VTC'2004 Spring, Milan, pp. 1-5, On CDrom, Mai, 2004.

- C. Vanstraceele, B. Geller, J. P. Barbot, J. M. Brossier, "An iterative phase synchronization scheme for general QAM constellations", Proc. ICC'2004, Paris, pp. 519-522, Vol. CT 08 Synchronization, June,2004.

- Brevet 1 : B. Geller, J.P. Barbot, J.M. Brossier, C. Vanstraceele, "Procédé de décodage itératif de codes blocs et dispositif décodeur correspondant", n° 04O6291, 10 Juin 2004.

- Brevet 2 : B. Geller, J.P. Barbot, J.M. Brossier, C. Vanstraceele, "Système de compensation de phase pour turbo décodeur", n° 04O6290, 10 Juin 2004.

- Brevet 3 : B. Geller, J.P. Barbot, J.M. Brossier, C. Vanstraceele, "Procédé d'estimation de la phase de données d'observation transmises sur un canal de transmission en modulation QAM", n°04P0441, 20 Septembre 2004.

Chapitre 2

Présentation de la chaine de transmission

Sommaire

2.1 Introduction

Ce chapitre a pour objet de présenter le modèle générique d'un chaîne de transmission de l'information. Nous n'avons pas ici pour objectif d'être exhaustif mais uniquement de préciser dans quel cadre ce travail de thèse a été mené. Ainsi nous présenterons les divers éléments constitutifs d'une chaîne de transmission en précisant ceux sur lesquels nos travaux ont porté. Nous introduirons également les notations qui seront utilisées dans l'ensemble de ce manuscript. L'orientation de ce travail ayant été fortement influencée par notre participation au projet MEDEA+ A106 INCA, une partie des éléments que recouvre ce chapitre sera précisée lors du chapitre 4.

Dans un premier temps nous présenterons les éléments constitutifs de la chaîne de transmission. Dans un second temps nous nous attarderons sur un type particulier de

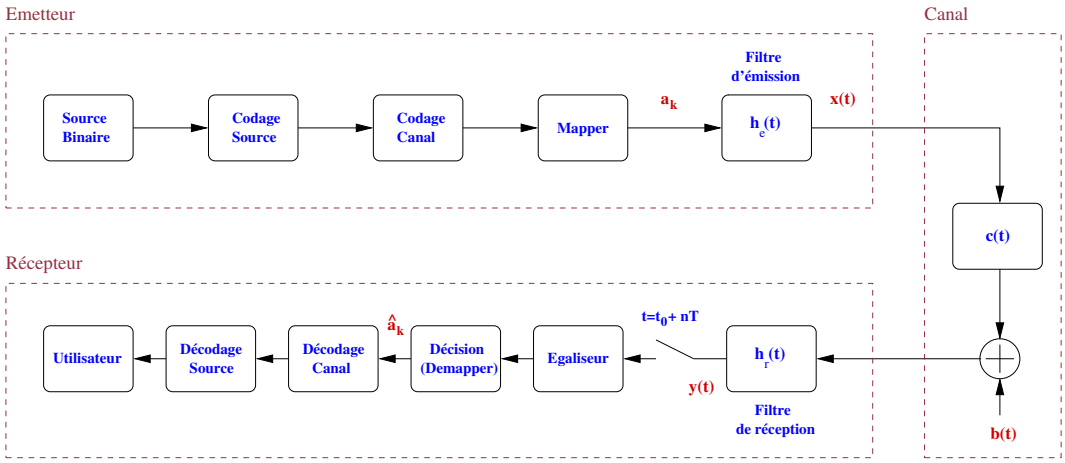


Figure 2.1: Système de transmission numérique.

modulation numérique, la Modulation d'Amplitude en Quadrature (MAQ). Ainsi, au risque de s'écarter du caractère généraliste de ce chapitre, cette partie présentera la modulation utilisée comme véhicule de l'information numérique à travers le canal de transmission. Nous profiterons également de cette partie pour présenter la notion de "demappeur souple" que nous utiliserons dans la suite de ce travail. Enfin, la dernière partie de ce chapitre précisera la notion de transmission multiporteuse. Ainsi nous introduirons en premier lieu la modulation OFDM (Orthogonal Frequency Division Multiplexing) afin de pouvoir décrire ensuite la modulation DMT (Discrete Multitone Modulation). Ce type de transmission multiporteuse ayant été retenue dans la norme ADSL, elle sera à la base des transmissions de données numériques objet du chapitre 4 de ce travail de thèse.

2.2 Transmission sur un canal à bande limitée

Le schéma de la figure 2.1 rappelle la structure générale d'un système de transmission numérique. On ne représente pas la modulation et la démodulation supposées transparentes, et l'on raisonne en bande de base. Dans tout ce système, nous supposerons que les équipements électroniques constituant la chaîne de transmission sont parfaits.

- Codage source

Le codage source a essentiellement pour objet la diminution de la quantité d'information à transmettre (compression), qui consiste à éliminer toute redondance inutile dans le message et donc à maximiser l'entropie. Le chiffrement du message transmis est également une fonction du codage source.

- Codage canal

Ce codage a pour objet la minimisation du taux d'erreur binaire, obtenu par l'introduction de redondances utiles dans le message. Il est censé lutter contre l'effet du bruit aléatoire

$b(t)$ présent sur la chaine de transmission. Le problème du codage canal est développé dans le chapitre 2.

- Mapper

Le message numérique, en tant que suite de bits, est une grandeur abstraite. Pour transmettre ce message, il est donc nécessaire de lui associer une représentation physique, sous forme d'un signal électrique. C'est le rôle de la concaténation du mapper avec le filtre d'émission. A chaque mot binaire de m bits, le mapper associe de manière univoque un symbole noté a_k choisi parmi un alphabet de $M = 2^N$ valeurs. A la sortie du mapper, la notion de message binaire disparaît au profit du signal $a(t)$:

$$a(t) = \sum_k a_k \delta\left(t - kT\right) \tag{2.1}$$

- Filtre d'émission

On convolue $a(t)$ par une impulsion $h_e(t)$, pour émettre ainsi une suite continue de signaux. Typiquement, $h_e(t)$ est un filtre en racine de cosinus surélevé, ce qui pemet d'occuper une bande passante relativement limitée tout en facilitant le fonctionnement du récepteur (intersymbole). D'où :

$$x\left(t\right) = a\left(t\right) \otimes h_e\left(t\right) = \sum_k a_k h_e\left(t - kT\right) \tag{2.2}$$

- Modulation/démodulation porteuse unique (non représenté fig. 2.1)

Plusieurs types de modulations numériques peuvent être envisagés pour la transmissions du signal $x\left(t\right)$. Nous nous restreindrons volontairement au cas de la modulation QAM (Quadrature Amplitude Modulation) qui sera utilisé dans la suite de ce travail. Prenons le cas élémentaire d'une modulation monoporteuse. On sait que la multiplication de $x(t)$ par $\exp\left(j2\pi f_0 t\right)$ déplace le spectre de $x(t)$ autour de $\pm f_0$. Le signal d'information $x(t)$ module ainsi une porteuse sinusoïdale qui transporte l'énergie. En réception, le signal reçu est multiplié par la même sinusoïde : un simple filtrage passe-bas permet de récupérer le signal $x(t)$ de départ dans le cas idéal, c'est-à-dire si l'on suppose que la synchronisation est parfaite. Ces opérations étant transparentes par rapport aux traitements effectués dans le modem (cf 2.3.3), nous ne considérons que les opérations faites en bande de base. Le problème de la synchronisation sera abordé au chapitre 3.

- Canal de transmission

Le canal de transmission ramené en bande de base peut être considéré en première approximation comme un canal linéaire invariant de réponse impulsionnelle $c(t)$. Les perturbations radio-fréquences et la diaphonie peuvent être modélisées par l'ajout d'un bruit aléatoire $b(t)$. Les éléments constitutifs de la chaîne de transmission contribuent également à ce bruit.

- Filtre de réception et égaliseur

En réception, un filtre adapté de réponse impulsionnelle $h_r(t)$ permet de maximiser le rapport signal à bruit et donc de minimiser la probabilité d'erreur. Pour un canal idéal $c(t) = \delta(t)$ le filtre adapté est défini par (2.3).

$$h_r(t) = \overline{h_e}(-t) \tag{2.3}$$

Il faut aussi éliminer les interférences inter-symboles. En effet, à la réception, on souhaite récupérer la valeur des symboles émis a_k. Pour ce faire, si on prend un échantillon par période à la réception :

$$\begin{aligned} y(t_0 + nT) &= \sum_k a_k r(t_0 + (n-k)T) + b(t_0 + nT) \\ &= a_n r(t_0) + \sum_{k \neq 0} a_k r(t_0 + (n-k)T) + b(t_0 + nT) \end{aligned} \tag{2.4}$$

Dans cette expression, $r(t)$ désigne la concaténation des filtres de mise en forme et de réception avec le canal de transmission, comme le résume l'équation (2.5) :

$$r(t) = h_e(t) \otimes c(t) \otimes h_r(t) \tag{2.5}$$

t_0 étant le temps de propagation du canal. On voit que $y(t_0 + nT)$ dépend du symbole a_n mais aussi des symboles voisins : c'est ce que l'on appelle les interférences inter-symboles. Lorsque le canal est idéal, c'est-à-dire que $c(t) = \delta(t)$, le filtre de réception doit éliminer les Interférences Inter-Symboles (IIS ou ISI en anglais), autrement dit, il faut choisir $h_r(t)$ de sorte que :

$$r(t_0 + (n-k)T) = 0, \ \forall n \neq k \tag{2.6}$$

Ceci constitue le critère de Nyquist. Il permet la synthèse des filtres d'émission et de réception. Si on suppose $c(t) = \delta(t)$, qui est le cas d'un canal idéal, on doit avoir :

$$\sum_{n=-\infty}^{+\infty} \left| H_e\left(f - \frac{n}{T}\right) \right|^2 = \text{constante} \tag{2.7}$$

où $H_e(f)$ est la fonction de transfert du filtre de mise en forme. En particulier, le filtre dit en "racine de cosinus surélevé" remplit cette condition. Il est représenté figure 2.2. Le coefficient α est appelé "roll-off". Ce filtre est couramment utilisé dans les systèmes de transmissions classiques. Le roll-off facilite l'égalisation mais augmente la bande nécessaire pour la transmission du signal. Dans le cas de l'OFDM, $\alpha = 0$ et le filtre de mise en forme est alors rectangulaire.

Nous avons considéré jusqu'ici le cas où la limitation de bande est uniquement le fait des filtres d'émission et de réception. Rappelons que ces filtres, qui sont définis par le concepteur du système, ont un rôle sur la limitation de la bande occupée par le signal à

transmettre, essentiel notamment lorsque plusieurs utilisateurs doivent travailler sur des portions voisines du spectre en limitant les brouillages qu'ils s'apportent mutuellement.

En fait le canal n'est pas idéal i.e. $c(t) \neq \delta(t)$, donc il provoque lui aussi des interférences inter-symboles. Le rôle de l'égaliseur sera par conséquent, à partir de l'estimation de cette fonction de transfert, de minimiser l'interférence inter-symbole.

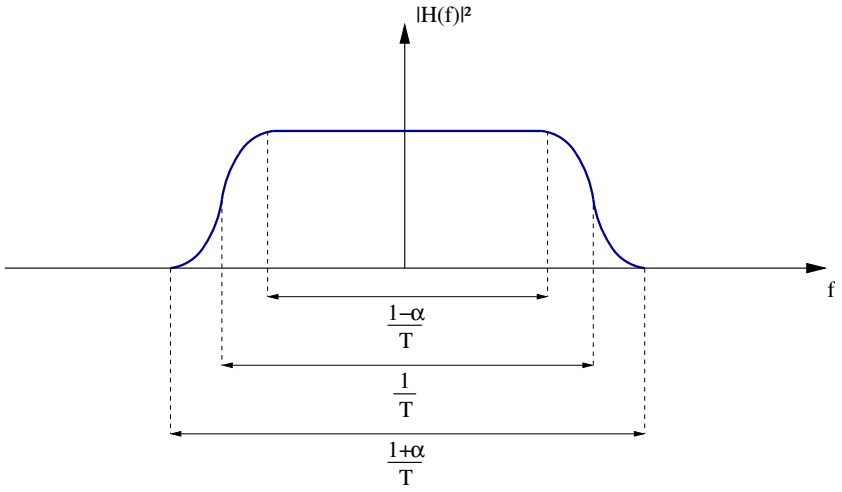


Figure 2.2: Filtre en cosinus surélevé.

- Décision

En sortie de l'égaliseur, on obtient des valeurs de symboles entachées d'erreurs à cause du bruit du canal et des imperfections des élements de la chaîne de transmission. Il faut alors décider quel symbole a été émis en fonction de l'échantillon reçu (demapping). La sortie du système de décision $\hat{a}_k$ ne peut être rigoureusement égale au symbole émis a_k : il y a toujours une probabilité d'erreur qui dépend du Rapport Signal sur Bruit (RSB).

- Décodage canal et décodage source

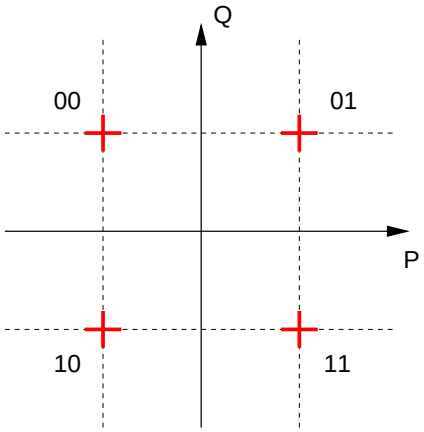
Ces blocs ont pour but de réaliser les fonctions inverses de celles décrites plus haut et en particulier le bloc décodage canal permettra de réduire la probabilité d'erreur sur les bits du message transmis.

2.3 Modulation d'Amplitude en Quadrature (MAQ)

Comme nous l'avions évoqué précédemment, la modulation MAQ sera exploitée dans la suite de ce travail. La présentation de quelques notions théoriques concernant la modulation MAQ (ou QAM en anglais) est essentielle pour la compréhension de la technique de modulation utilisée.

2.3.1 Utilisation des constellations de symboles MAQ

Dans un tel système de modulation, deux composantes en quadrature de phase d'une porteuse sont multipliées chacune par une valeur (paire de données p_k et q_k) prise parmi un ensemble fini de valeurs prédéfinies (constellation MAQ). Ensuite ces deux composantes en quadrature sont additionnées afin de construire le signal MAQ désiré (signal modulé de durée T secondes). Chaque paire de valeurs ou de coordonnées p_k et q_k peut être représentée par un point dans un repère bidimensionnel. En général, pour des considérations pratiques, le nombre de points dans une constellation est une puissance de deux. La figure 2.3 donne un exemple d'une constellation MAQ.



P : Phase (partie réelle $p_k$du symbole MAQ)

Q: Quadrature (partie imaginaire q_k du symbole MAQ)

Symbole MAQ : $a_k = p_k + jq_k$

Figure 2.3: Constellation MAQ.

Notons $M = 2^N$ le nombre de symboles de la constellation MAQ ; N représentant le nombre de bits associé à un symbole de la constellation MAQ. Dans notre exemple de la figure 2.3 , N vaut 2, M est donc égal à 4.

Le débit de transmission D vaut alors :

$$D = \frac{N}{T} \text{ bit/s} \tag{2.8}$$

La performance d'une modulation de type MAQ, en présence d'un bruit blanc additif de type Gaussien, dépend de la distance entre les points de la constellation utilisée. Les meilleurs résultats sont dons obtenus avec un type de constellation dont les symboles sont répartis de manière judicieuse dans le plan (I-Q). La constellation précédente est une constellation de type carrée très performante, mais il existe aussi des constellations de type circulaire (symboles situés sur un cercle de rayon égal à 1), moins performantes en terme de probabilité d'erreur en fonction du RSB (constellations PSK), mais plus robustes sur les canaux variables de type radio car indépendante de l'amplitude.

2.3.2 Principe théorique de la modulation MAQ

Soit f_0 la fréquence porteuse sur laquelle on veut transmettre le symbole MAQ désiré de composantes p_k et q_k. La composante p_k va moduler en amplitude le cosinus et la

composante q_k la partie en quadrature en sinus (p_k et q_k représentent les coordonnées dans le plan complexe du symbole MAQ). Le signal MAQ résultant après addition des deux parties en phase et en quadrature va être :

$$s_k(t) = p_k cos\,(2\pi f_0 t) - q_k sin\,(2\pi f_0 t) \tag{2.9}$$

ce signal étant présent pendant une durée T appelée durée symbole. Ou peut également écrire (2.9) sous cette forme :

$$s_k(t) = r_k cos\,(2\pi f_0 t + \phi_k) \tag{2.10}$$

r_k et ϕ_k étant les coordonnées polaires du symbole MAQ. Si l'on considère une transmission en modulation MAQ de plusieurs symboles ayant chacun une durée T, le signal transmis dans le temps devient alors :

$$s(t) = \sum_k \left[p_k cos\,(2\pi f_0 t) - q_k sin\,(2\pi f_0 t)\right] \Pi\,(t - kT) \tag{2.11}$$

où $\Pi\,(t)$ est la porte temporelle de largeur T ($\Pi\,(t) = 1$ pour $t \in [0, T]$, $\Pi\,(t) = 0$ ailleurs).

2.3.3 Notion d'enveloppe complexe du signal

Soit un signal réel $s\,(t) = Re\,(s_a\,(t))$ où $s_a\,(t)$ est appelé signal analytique. Il y a une relation biunivoque en définissant le signal $s_a\,(t)$ comme suit :

$$s_a\,(t) = s\,(t) + j\check{s}\,(t) \tag{2.12}$$

où $\check{s}\,(t)$ est la transformée de Hilbert du signal $s\,(t)$, et donc :

$$s_a\,(t) = s\,(t) \otimes \left(\delta\,(t) + j\frac{1}{\pi t}\right) \tag{2.13}$$

On a alors :

$$S_a\,(f) = \begin{cases} 2S\,(f) & si\ f > 0 \\ S(f) & si\ f = 0 \\ 0 & si\ f < 0 \end{cases} \tag{2.14}$$

$s_a\,(t)$ peut se mettre sous la forme :

$$s_a\,(t) = x\,(t)\, e^{j2\pi f_0 t} \tag{2.15}$$

où $x\,(t)$ est alors appelé "enveloppe complexe du signal" et représente le signal en bande de base (non modulé). Cette notation permet de s'affranchir de la modulation dans l'étude d'un système de transmission. Dans le cas de la modulation d'amplitude en quadrature, l'enveloppe complexe est donc :

$$x(t) = \sum_k a_k \Pi\,(t - kT) \tag{2.16}$$

et on a :

$$s(t) = Re\left(x(t)e^{j2\pi f_0 t}\right) \tag{2.17}$$

Cette représentation est particulièrement interessante car la modulation peut alors être considérée comme opération transparente du point de vue de la récéption.

2.3.4 Modulateur et démodulateur MAQ réel

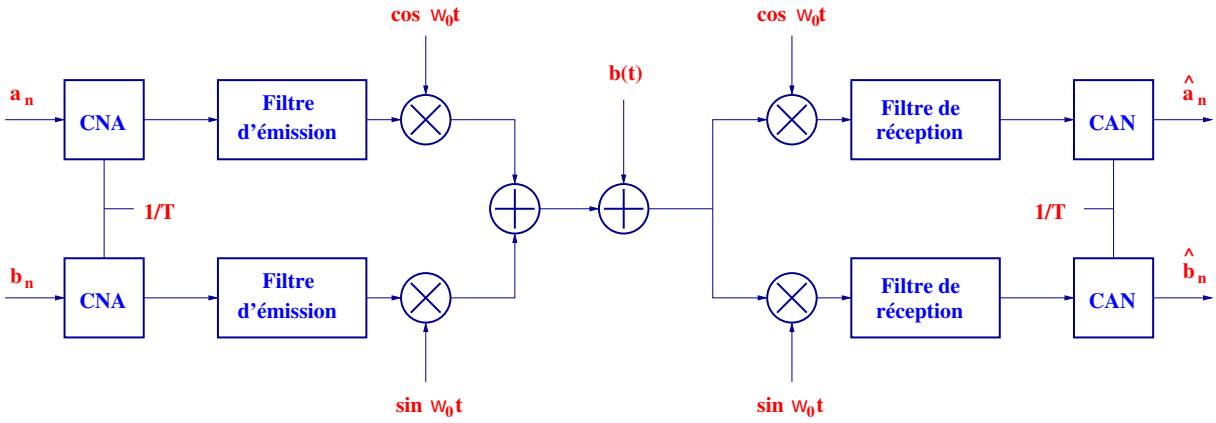


Figure 2.4: Modulateur et démodulateur MAQ.

Dans un système MAQ, les deux Convertisseurs Numérique-Analogique (CNA) délivrent une impulsion pour chaque symbole MAQ au débit de $1/T$ impulsions/sec. Dans le cas d'un filtre de mise en forme à réponse rectangulaire de durée T, dans le domaine des fréquences le spectre de ces impulsions a une étendue infinie en $\sin(x)/x$ (sinus cardinal). En effet, la transformée de Fourier d'une porte dans le domaine temporel est un sinus cardinal dans le domaine des fréquences. La limitation de la bande du canal de transmission nécessite, dans la réalité, un filtrage pour adapter le spectre de la porteuse modulée en MAQ à la bande passante du canal. C'est pourquoi la mise en forme sera assurée dans la plupart des cas par un demi Nyquist (filtre en racine de cosinus surelevé) dont l'occupation spectrale est limitée et qui permettra d'annuler l'IIS lorsque le canal est idéal. L'enveloppe complexe du signal devient alors :

$$x(t) = \sum_k a_k h_e\left(t - kT\right) \tag{2.18}$$

où $h_e(t)$ est la réponse impulsionnelle du filtre de mise en forme utilisé.

2.3.5 Performance en présence de bruit gaussien

Soit un bruit additif, blanc, gaussien et centré de variance σ^2. On considère le cas où il n'y a pas de codage canal. Un échantillon de bruit b, comme le montre la figure 2.5,

a pour effet de produire à la réception un point O' qui n'appartient pas à la constellation. Naturellement, pour minimiser les risques d'erreur, on recherchera un point de la constellation à distance euclidienne la plus faible possible par rapport au point reçu.

Si l'on suppose que la modulation comporte $M = 2^N$ points $\{O_1, O_2, \cdots, O_M\}$ auxquels sont associées des amplitudes $\{A_1, A_2, \cdots, A_M\}$ avec des probabilités d'apparition de chaque symbole $\{p_1, p_2, \cdots, p_M\}$, la puissance moyenne est alors :

$$S^2 = \sum_{i=1}^{M} p_i A_i^2 \tag{2.19}$$

et on définit le rapport signal à bruit par :

$$SNR = \frac{S^2}{\sigma^2} \tag{2.20}$$

Considérons le cas d'une MAQ à 2 état (2-QAM). En appelant A la demi-distance entre les deux points de la constellation, la probabilité d'erreur sur le symbole émis est :

$$P_{es} = P\left(b > A\right) = \int_A^{+\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} dx \tag{2.21}$$

Si l'on pose $\text{erfc}\left(x\right) = \frac{2}{\sqrt{\pi}} \int_x^{+\infty} e^{-u^2} du$, on a :

$$P_{es} = \frac{1}{2}\text{erfc}\left(\frac{A}{\sigma\sqrt{2}}\right) \tag{2.22}$$

Dans le cas d'une 2-QAM (ou BPSK), la probabilité d'erreur par symbole est donnée par :

$$P_{es} = \frac{1}{2}\text{erfc}\left(\frac{S}{\sigma\sqrt{2}}\right) \tag{2.23}$$

Elle est égale à la probabilité d'erreur par bit P_{eb}. En posant la densité spectrale monolatérale du bruit N_0 et B la largeur de bande équivalente de bruit, telle que $\sigma^2 = \frac{1}{2} N_0 B$ et $E_b = \frac{S^2}{N}\frac{1}{B}$ l'énergie associée à un bit, on a :

$$P_{eb} = \frac{1}{2}\text{erfc}\left(\sqrt{\frac{E_b}{N_0}}\right) \tag{2.24}$$

De même, dans le cas d'une 4-QAM, à fort RSB la probabilité d'erreur sur un bit est donnée par :.

$$P_{eb} = \frac{1}{2}\text{erfc}\left(\frac{A}{\sigma\sqrt{2}}\right) = \frac{1}{2}\text{erfc}\left(\frac{S}{2\sigma}\right) = \frac{1}{2}\text{erfc}\left(\sqrt{\frac{E_b}{N_0}}\right) \tag{2.25}$$

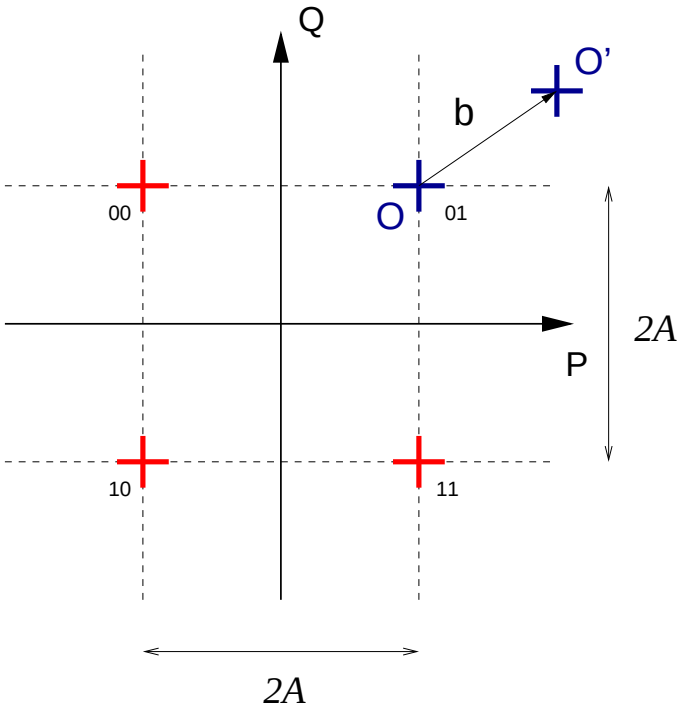


Figure 2.5: Effet du bruit à la reception.

Le but sera bien entendu de limiter ce taux d'erreur à la reception tout en optimisant le débit. Pour cela, il faut bien sur adapter la constellation utilisée aux propriétés du canal. La figure 2.6 représente les taux d'erreur binaire pour quelques constellations MAQ carrées, en fonction du rapport entre l'énergie binaire et la densité spectrale du bruit. On constate une dégradation du taux d'erreur binaire lorsque le nombre de points de la constellation augmente.

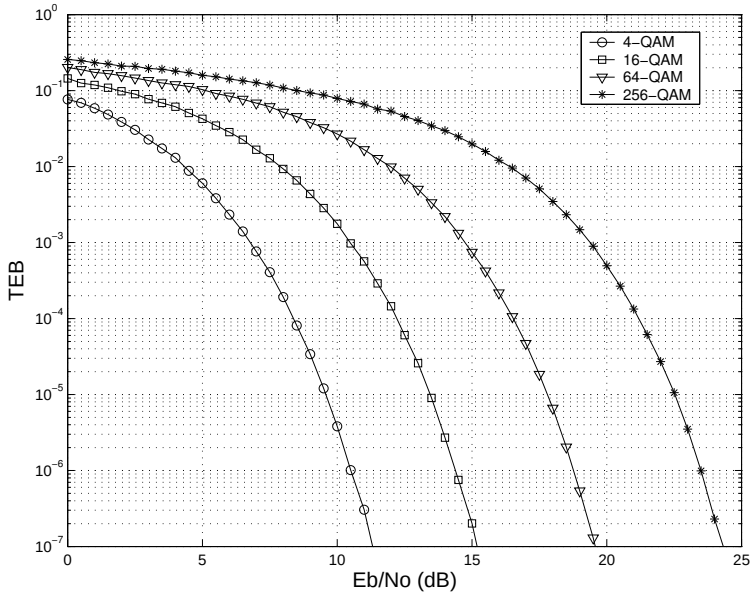


Figure 2.6: TEB pour plusieurs constellations.

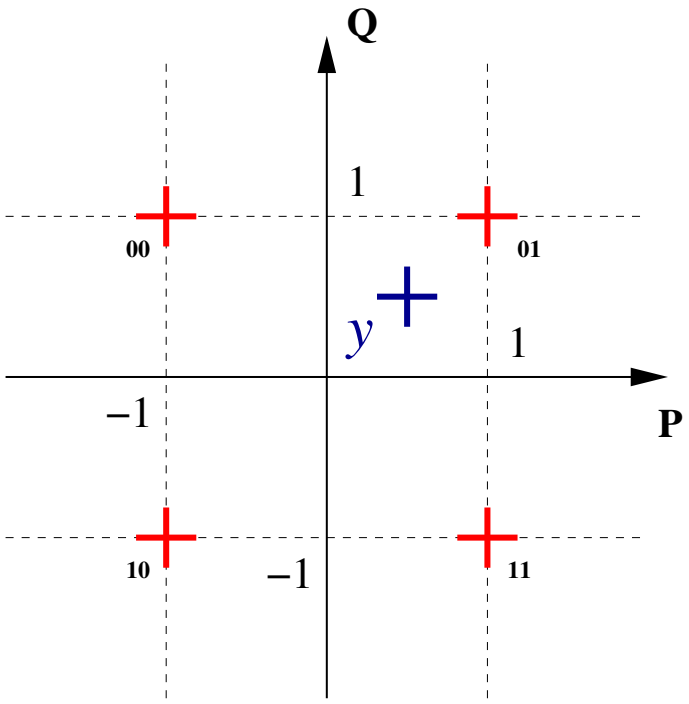


Figure 2.7: Exemple pour le calcul de log-vraisemblance.

2.3.6 Demappeur souple

Dans certains cas, le décodeur canal utilisé dans le récepteur possède ce que l'on appelle une "entrée souple" (ou "soft input"). Cela suppose qu'en plus de la décision sur les bits reçus, on possède une information sur la fiabilité de cette décision. Il revient alors au demappeur de fournir cette information supplémentaire. Pour cela, on calcule ce que l'on appelle la log-vraisemblance (en anglais log-likelihood ratio notée LLR), qui est définie par l'équation suivante :

$$\Lambda\left(b_i^k\right) = \ln\left[\frac{P\left(b_i^k = 1/y_k\right)}{P\left(b_i^k = 0/y_k\right)}\right] \tag{2.26}$$

où y_k est le symbole reçu et b_i^k est le $i^{ème}$ bit associé à la décision sur ce symbole. Cette grandeur est du signe correspondant à la décision dure sur le bit considéré et sa valeur absolue représente une mesure de la fiabilité de la décision. Or, en considérant tous les symboles de la constellation, on a :

$$P\left(b_i^k = 1/y_k\right) = \sum_{Q_m/q_i^m=1} P\left(a_k = Q_m/y_k\right) \tag{2.27}$$

et :

$$P\left(b_i^k = 0/y_k\right) = \sum_{Q_m/q_i^m=0} P\left(a_k = Q_m/y_k\right) \tag{2.28}$$

avec a_k le symbole émis, $\{Q_1, Q_2, \cdots, Q_M\}$ les symboles d'une constellation M-QAM ($M = 2^N$) et q_i^m le $i^{ème}$ bit de l'étiquetage du symbole Q_m. On a donc :

$$\Lambda\left(b_i^k\right) = \ln\left[\frac{\sum_{Q_m/q_i^m=1} P\left(a_k = Q_m/y_k\right)}{\sum_{Q_m/q_i^m=0} P\left(a_k = Q_m/y_k\right)}\right] \tag{2.29}$$

Or, d'aprés la loi de Bayes et si l'on considère les symboles équiprobables, on a :

$$\Lambda\left(b_i^k\right) = \ln\left[\frac{\sum_{Q_m/q_i^m=1} P\left(y_k/a_k = Q_m\right)}{\sum_{Q_m/q_i^m=0} P\left(y_k/a_k = Q_m\right)}\right] \tag{2.30}$$

et on sait que du fait de l'hypothèse de bruit gaussien :

$$P\left(y_k/a_k = Q_m\right) = \frac{1}{\pi\sigma^2}\exp\left(-\frac{\left|y_k - Q_m\right|^2}{\sigma^2}\right) \tag{2.31}$$

avec σ^2 la variance de ce bruit gaussien présent sur le canal.
Donc, finalement :

$$\Lambda\left(b_i^k\right) = \ln\left[\frac{\sum_{Q_m/q_i^m=1} \exp\left(-\frac{\left|y_k - Q_m\right|^2}{\sigma^2}\right)}{\sum_{Q_m/q_i^m=0} \exp\left(-\frac{\left|y_k - Q_m\right|^2}{\sigma^2}\right)}\right] \tag{2.32}$$

Par exemple, considérons l'utilisation d'une constellation 4-QAM de la figure 2.7, et la réception d'un symbole bruité y. Si l'on considère que le symbole y a pour coordonnées complexes $y = 0,5 + 0,5j$, alors la log-vraisemblance du premier bit de y pourra se calculer par :

$$\Lambda(b_1) = ln \frac{\exp\left(-\frac{|0,5+0,5j-(-1-j)|^2}{\sigma^2}\right) + \exp\left(-\frac{|0,5+0,5j-(1-j)|^2}{\sigma^2}\right)}{\exp\left(-\frac{|0,5+0,5j-(-1+j)|^2}{\sigma^2}\right) + \exp\left(-\frac{|0,5+0,5j-(1+j)|^2}{\sigma^2}\right)}$$

Si on considère $\sigma^2 = 1$ par exemple, on obtient $\Lambda(b_1) = -2$. Le signe nous indique que le bit en question est de valeur 0 et la valeur absolue indique la fiabilité de ce bit.

2.4 Modulations multiporteuses

2.4.1 La modulation OFDM

2.4.1.1 Principe

La modulation OFDM (Orthogonal Frequency Division Multiplexing) [WE71][HWK00][Bin90] est une modulation de type multiporteuse utilisant comme données des symboles de constellation MAQ. Le terme de modulation multiporteuse désigne une modulation qui utilise conjointement plusieurs porteuses en fréquence comme supports de transmission des données. Un type de constellation MAQ est choisi pour le « mapping » des données. L'écart entre chaque porteuse est égal Δf. Cet écart est choisi en tenant compte des recouvrements spectraux. Le signal transmis pendant la durée du $k^{ème}$ symbole (durée T) sur la $n^{ième}$ porteuse de fréquence f_n est donné par :

$$s_{n,k}(t) = p_{n,k} cos\left(2\pi f_n t\right) - q_{n,k} sin\left(2\pi f_n t\right) \tag{2.33}$$

En considérant la transmission de K symboles successifs, le signal devient :

$$s_n(t) = \sum_k \left[p_{n,k} cos\left(2\pi f_n t\right) - q_{n,k} sin\left(2\pi f_n t\right)\right] \Pi\left(t - kT\right) \tag{2.34}$$

Dans le cas d'une modulation multiporteuse, l'expression du signal résultant de la superposition de N_p porteuses est donné par (2.35).

$$s(t) = \sum_{k=0}^{K} \sum_{n=1}^{N_p} \left[p_{n,k} cos\left(2\pi f_n t\right) - q_{n,k} sin\left(2\pi f_n t\right)\right] \Pi\left(t - kT\right) \tag{2.35}$$

où n est l'indice des fréquences porteuses et k l'indice temporel. L'enveloppe complexe du signal transmis est donc :

$$x(t) = \sum_{k=0}^{K} \sum_{n=1}^{N_p} a_{n,k} h_n\left(t - kT\right) \tag{2.36}$$

avec $h_n(t) = e^{j2\pi n\Delta ft}\Pi(t)$ où Δf est la distance entre deux porteuses consécutives et donc $f_n = n\Delta f$. Si la transmission se fait autour d'une fréquence porteuse f_0, le signal transmis au final sera donné par :

$$s(t) = Re\left(x(t)e^{j2\pi f_0 t}\right) \tag{2.37}$$

Au niveau spectral, les différentes porteuses se superposent sur la largeur de bande du canal de transmission. La densité spectrale de puissance résultante est alors constante sur la largeur de la bande du canal de transmission.

En réception, le processus de démodulation peut être formalisé par l'équation suivante (basé sur la propriété d'orthogonalité des porteuses) :

$$\int_{kT}^{(k+1)T} s(t)e^{j2\pi f_n t}dt = \frac{T}{2}a_{n,k} \tag{2.38}$$

En considérant que chaque porteuse transporte un symbole pris dans une constellation QAM (même constellation pour toutes les porteuses) contenant 2^m points durant un temps de T secondes, le débit est alors donné par :

$$D = N\frac{m}{T} \quad \text{bit/s} \tag{2.39}$$

L'OFDM possède deux avantages principaux : pour générer (respectivement démoduler) le signal OFDM on utilise la transformée de Fourier rapide inverse notée IFFT (respectivement directe notée FFT), ce qui permet un traitement simple, avec une excellente efficacité spectrale. L'OFDM permet une limitation de l'interférence inter-symbole entre les symboles QAM grâce à une transmission parallèle des bits qui allonge la durée des symboles. Le paragraphe expose le principe généralement utilisé pour le calcul d'une trame OFDM.

2.4.1.2 Principe du calcul d'un signal OFDM

Nous allons exposer une méthode [CMBL00] simplifiant la génération d'un signal OFDM. Cette méthode permet de générer un signal OFDM de $2N$ échantillons par l'utilisation d'une *TFDI* de longueur N au lieu de celle classique de longueur $2N$.

L'approche de cette méthode est la suivante :

• la Transformée Discrète de Fourier Inverse (*TFDI*) donnant un vecteur réel de longueur $2N$ peut être calculée à partir de deux vecteurs réels de longueur N,

• la *TFDI* donnant deux signaux réels de longueur N peut être calculée en utilisant un seul bloc de calcul de *TFDI* de longueur N.

Soit $\mathbf{C}$ le vecteur complexe de $2N$ coordonnées à symétrie hermitienne représentant le signal à transmettre dans le domaine fréquentiel. La composante $\mathbf{C}_{N+k}$ $(k \geq 0)$ est le nombre complexe $a_k + jb_k$ associé au symbole QAM sur la $k^{ième}$ porteuse. La composante C_{N-k} $(k \geq 0)$ est choisie comme étant le nombre complexe $a_k - jb_k$. Soit x_q le vecteur réel de longueur $2N$ obtenu par *TFDI* du vecteur C_k, on a alors :

$$x_q = \sum_{k=0}^{2N-1} C_k . \exp\left(\frac{j2\pi kq}{2N}\right) \tag{2.40}$$

x_q représentant le vecteur discret constitué des $2N$ échantillons du signal DMT réel. Les échantillons pairs et impairs peuvent être alors réécrits de la manière suivante :

$$\begin{aligned} x_{2p} &= \sum_{k=0}^{2N-1} C_k . \exp\left(\frac{j2\pi k2p}{2N}\right) \\ &= \sum_{k=0}^{N-1} C_k . \exp\left(\frac{2j\pi kp}{N}\right) + \sum_{k=N}^{2N-1} C_k . \exp\left(\frac{2j\pi kp}{N}\right) \\ &= \sum_{k=0}^{N-1} \left(C_k + C_{k+N}\right) . \exp\left(\frac{2j\pi kp}{N}\right) \end{aligned} \tag{2.41}$$

d'où :

$$\{x_{2p}\} = TFDI_N \{C_k + C_{k+N}\} \tag{2.42}$$

et de même :

$$\begin{aligned} x_{2p+1} &= \sum_{k=0}^{2N-1} C_k . \exp\left(\frac{2j\pi k(2p+1)}{2N}\right) \\ &= \sum_{k=0}^{N-1} C_k . \exp\left(\frac{2j\pi kp}{N}\right) . \exp\left(\frac{j\pi k}{N}\right) + \sum_{k=N}^{2N-1} C_k . \exp\left(\frac{2j\pi kp}{N}\right) . \exp\left(\frac{j\pi k}{N}\right) \\ &= \sum_{k=0}^{N-1} \left(C_k - C_{k+N}\right) . \exp\left(\frac{2j\pi kp}{N}\right) . \exp\left(\frac{j\pi k}{N}\right) \end{aligned} \tag{2.43}$$

d'où :

$$\{x_{2p+1}\} = TFDI_N \{(C_k - C_{k+N}) \exp(j\pi k/N)\} \tag{2.44}$$

Les N échantillons impairs du signal OFDM peuvent donc être calculés à partir d'une $TFDI$ de N points des termes $(C_k - C_{k+N}).exp(j\pi k/N)$ et les échantillons pairs du signal OFDM peuvent être calculés à partir d'une $TFDI$ de N points des termes $C_k + C_{k+N}$. Dans notre cas, les échantillons x_q sont réels (signal OFDM réel) et correspondent à une symétrie hermitienne des échantillons complexes :

$$C_{k+N} = \bar{C}_{N-k} \tag{2.45}$$

Ainsi les échantillons pairs et impairs du signal OFDM réel peuvent être calculés à partir de N coefficients C_k (k allant de 0 à $N-1$) avec les expression (2.46) et (2.47).

$$\{x_{2p}\} = TFDI_N \left\{C_k + \bar{C}_{N-k}\right\} \tag{2.46}$$

$$\{x_{2p+1}\} = TFDI_N \left\{ (C_k - \bar{C}_{N-k}) . \exp\left(\frac{j\pi k}{N}\right) \right\} \tag{2.47}$$

Le fait que les échantillons x_{2p}et x_{2p+1} soient réels implique que les deux $TFDI_N$ peuvent être calculées simultanément par un seul bloc $TFDI$ de N points complexes. En effet si $\{a_q\}$=$TFDI\{A_k\}$ et $\{b_q\}$=$TFDI\{B_k\}$ alors par linéarité on a :

$$\{a_q + jb_q\} = TFDI\{A_k + jB_k\} \tag{2.48}$$

d'où :

$$\{x_{2p} + jx_{2p+1}\} = TFDI_N\left\{(C_k + \bar{C}_{N-k}) + (C_k - \bar{C}_{N-k}).\exp(\frac{j\pi k}{N}).\exp(j\frac{\pi}{2})\right\} \tag{2.49}$$

et donc :

$$\{x_{2p}\} = Re\left[TFDI_N\left\{(C_k + \bar{C}_{N-k}) + (C_k - \bar{C}_{N-k}).\exp(\frac{j\pi k}{N}).\exp(j\frac{\pi}{2})\right\}\right] \tag{2.50}$$

et aussi :

$$\{x_{2p+1}\} = Im\left[TFDI_N\left\{(C_k + \bar{C}_{N-k}) + (C_k - \bar{C}_{N-k}).\exp(\frac{j\pi k}{N}).\exp(j\frac{\pi}{2})\right\}\right] \tag{2.51}$$

La figure 2.8 donne de manière synthétique une vision des calculs effectués (prétraitement des données) avec $2N\Delta f$ la fréquence d'échantillonnage (Δf représente l'écart de fréquence entre chaque porteuse de la modulation OFDM).

Cette méthode requiert donc un nombre d'opérations pour la $TFDI$ (ou $IFFT$) de l'ordre de $N\log_2 N$ au lieu de $2N\log_2 2N$, ce qui permet un gain de calcul très important. Une table contenant N valeurs des fonctions trigonométriques cosinus et sinus est également nécessaire (N constantes multiplicatives).

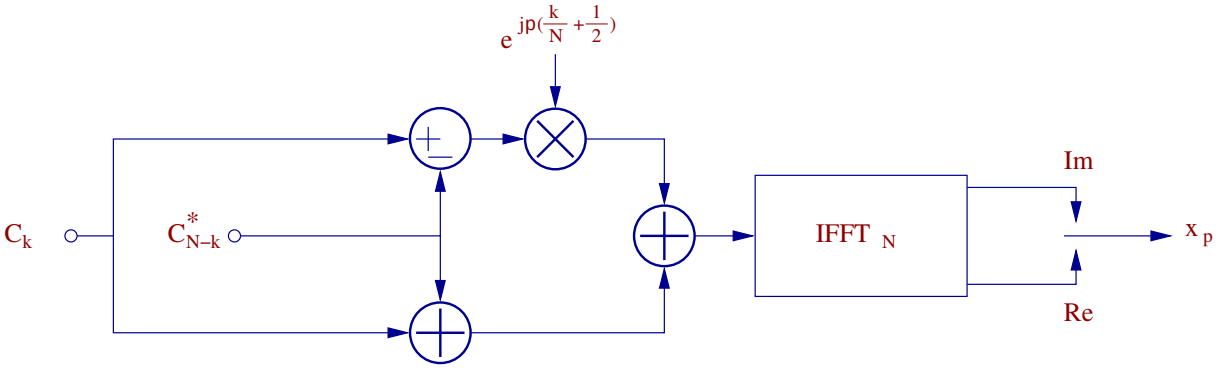


Figure 2.8: Génération d'un signal OFDM.

2.4.2 La modulation DMT

La modulation DMT (Discrete Multitone Modulation), comme la modulation OFDM est une modulation multiporteuse. Toutefois, dans le cas de la modulation DMT la constellation associée à chaque porteuse n'est pas forcément toujours la même. Le choix de la

constellation est réalisé en tenant compte du rapport signal à bruit pour la porteuse considérée (principe du " water filling " de Shannon [Sha48]). Si pour une fréquence le bruit est important, une configuration de constellation avec peu de points sera alors choisie ; a contrario pour des fréquences porteuses robustes au bruit, des constellations formées d'un grand nombre de points seront retenues. On réalise de ce fait une adaptation du débit de chaque sous-porteuse en fonction de sa capacité. La génération du signal DMT, dans le cas du projet modem VDSL, est réalisée en employant la technique présentée au 2.4.1.2.

2.4.2.1 Principe de base du mapping et demapping des données en modulation DMT

Le principal avantage de la modulation DMT est de permettre d'allouer un nombre variable de bits pour chaque sous-porteuse en considérant la caractéristique rapport signal sur bruit autour de la fréquence porteuse considérée [Kal89]. L'allocation du nombre de bits suivant chaque porteuse se fait par voie de retour au moment de l'initialisation du modem VDSL. De plus, la distribution des bits est faite d'une manière adaptative pour tenir compte de changements éventuels des caractéristiques du canal de transmission au cours du temps. Cette procédure porte le nom de *bit loading*. Cette phase constitue une partie du mapping (mise en forme) des données à transmettre. Elle permet d'adapter au mieux le système au canal de transmission et ainsi d'optimiser le débit pour se rapprocher de la capacité du canal.

Procédure d'allocation du nombre de bits : bit swapping/loading L'allocation du nombre de bits suivant la porteuse i du signal DMT peut se faire par ce calcul [SCS99][Bin00]:

$$b(i) = \log_2(1 + \frac{RSB(i)}{\Gamma}) \tag{2.52}$$

où Γ est une constante déterminée par le taux d'erreur (BER, *Bit Error Rate*) que l'on s'autorise et RSB(i) le rapport signal à bruit de chaque porteuse i.

Considérant cette allocation du nombre de bits par porteuse, le débit d'une paire torsadée en cuivre (Réseau Téléphonique) est donnée par :

$$C = \frac{1}{T}\sum_{i=1}^{N} b(i) = \frac{1}{T}\sum_{i=1}^{N} \log_2(1 + \frac{RSB(i)}{\Gamma}) \tag{2.53}$$

Cette équation occulte le fait que l'utilisation de codes correcteurs dans l'émetteur permet d'obtenir un gain en terme de RSB. En considérant que le fait d'utiliser un code correcteur d'erreur permet d'obtenir un gain g et que le système de transmission travaille avec une marge de puissance de m (marge de sécurité), l'équation (2.52) devient alors :

$$b(i) = \log_2(1 + \frac{RSB(i)}{\Gamma}.\frac{g}{m}) \tag{2.54}$$

On peut voir sur la figure 2.9 un exemple d'allocation des bits sur les porteuses en fonction de la distribution des RSB dans un système exploitant une modulation DMT.

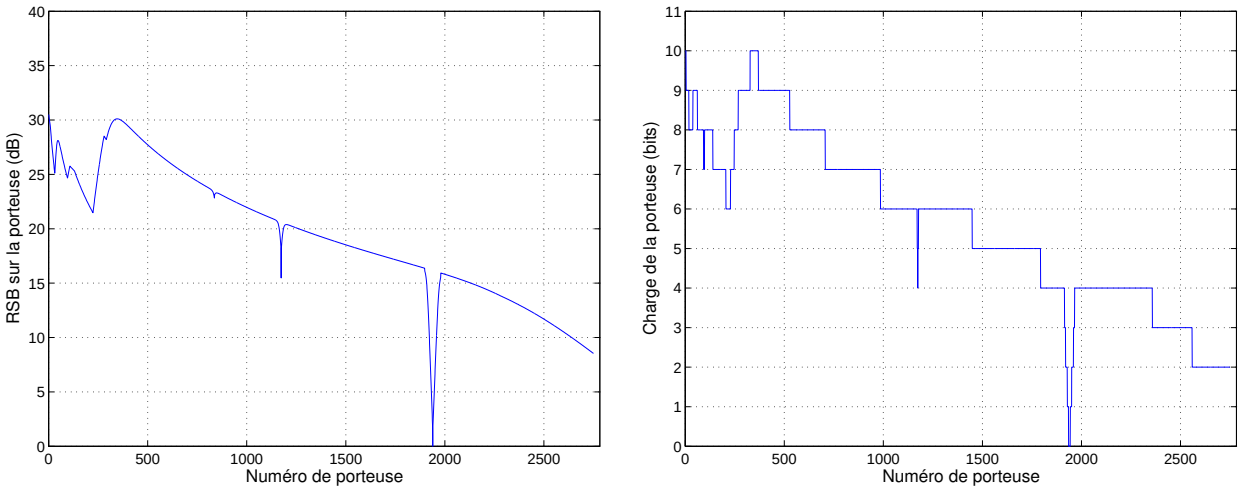


Figure 2.9: Exemple d'allocation des bits sur les porteuses en fonction du RSB.

Configuration des constellations QAM Les constellations utilisées pour mapper les données ont une dénomination particulière en fonction de la longueur en nombre de bits du symbole à mapper. Soit N le nombre de bits du symbole, ce symbole fera alors partie de la constellation 2^N-QAM. Ces constellations QAM sont de trois types : spécifiques (constellations 2-QAM et 8-QAM (1 bit et 3 bits)), carrées (valeurs paires du nombre de bits : 4-QAM, 16-QAM, 64-QAM, 256-QAM, 1024-QAM, 4096-QAM et 16384-QAM) et en croix (valeurs impaires du nombre de bits : 32-QAM, 128-QAM, 512-QAM, 2048-QAM, 8192-QAM, 32768-QAM). Ces types de constellations QAM ont été choisies par la *VDSL Alliance* (consortium de plusieurs sociétés pour le projet VDSL). Les figures 2.10 à 2.12 donnent des exemples de ces différentes constellations, où le nombre associé à chaque point est la conversion en décimal de la séquence binaire associée.

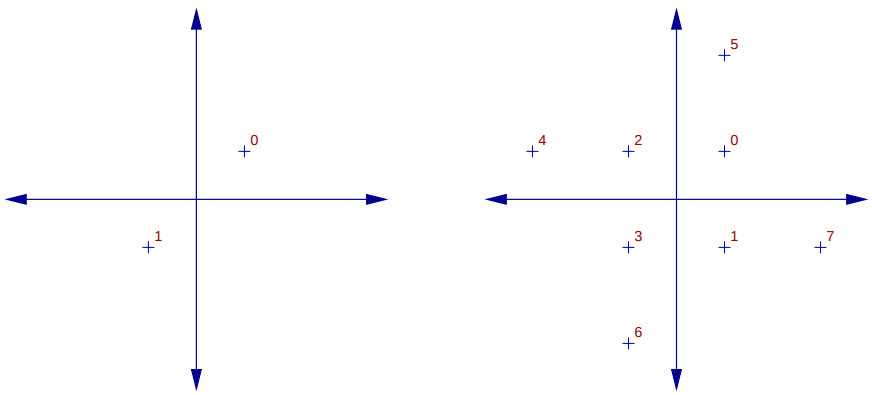


Figure 2.10: Constellations spécifiques.

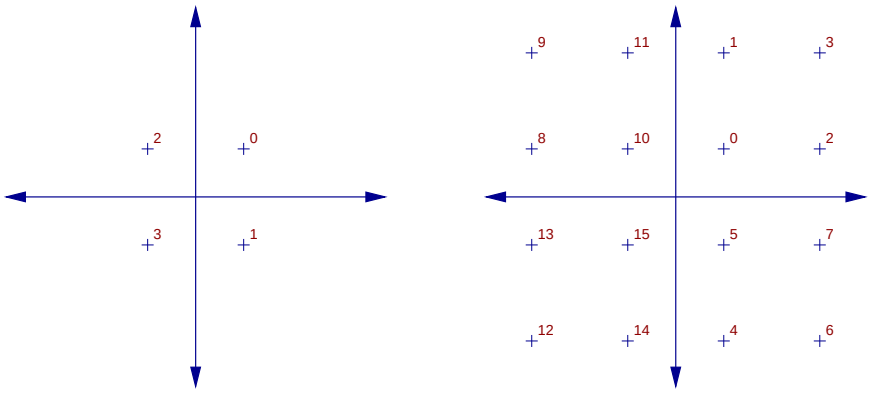


Figure 2.11: Constellations carrées (n pair).

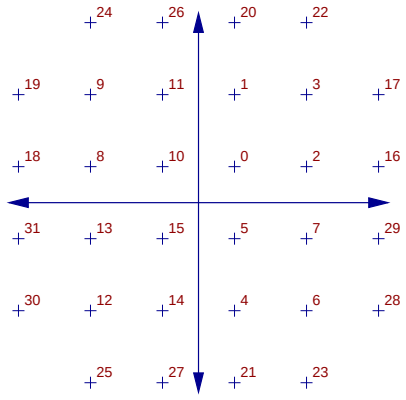


Figure 2.12: Constellations en croix (n impair).

2.4.2.2 Extension Cyclique

Le fait de rajouter un préfixe cyclique [Iss] permet de circulariser la Transformée de Fourier à la réception. La conséquence est la limitation de l'interférence entre symboles OFDM adjacents si le préfixe permet d'absorber la durée de la réponse impulsionnelle du canal. Cela facilite donc l'opération d'égalisation à la réception ainsi que la synchronisation sur les symboles DMT (ou OFDM).

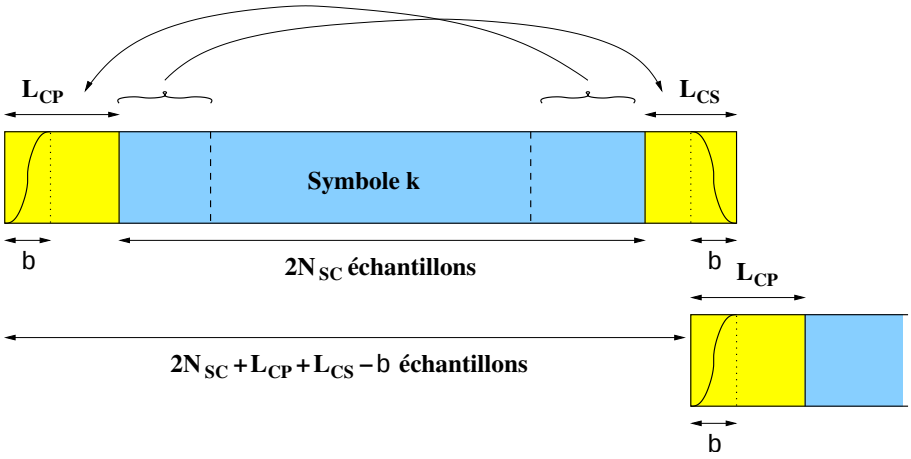


Figure 2.13: Extension cyclique.

Les L_{CP} derniers échantillons du signal DMT seront recopiés en début de séquence comme préfixe cyclique. Les L_{CS} premiers échantillons seront recopiés en fin de séquence comme suffixe cyclique.

Les β premiers échantillons du préfixe et les β derniers échantillons du suffixe seront utilisés pour modifier l'enveloppe du signal transmis (*windowing*). Les β premiers et derniers échantillons de deux trames consécutives se chevauchent.

L'extension cyclique totale est définie comme $L_{CE} = L_{CP} + L_{CS} - \beta$. Le schéma 2.13 résume les opérations effectuées.

2.5 Conclusion

Dans ce chapitre nous avons rappelé quelques bases élémentaires des communications numériques. Nous avons également précisé certaines parties exploitées dans ce travail de thèse, à savoir les modulations MAQ et l'utilisation des transmissions multiporteuses de type DMT. L'apport majeur des transmissions DMT est de pouvoir mettre en oeuvre le principe de "water filling" proposé par Shannon, autorisant ainsi une optimisation de l'utilisation de la capacité globale du canal de transmission. Ainsi le choix de la DMT se justifie t-il dans le cadre des systèmes ADSL où le canal de transmission est peu variable dans le temps. Dans ce cas, les sous-canaux présentant un faible rapport signal sur bruit, et donc une faible capacité de transmission, peuvent être identifiés lors de la phase d'initialisation des modems ADSL. L'association MAQ et DMT est également candidate potentielle pour l'élaboration de la norme VDSL. Elle sera donc retenue comme principe de modulation et de transmission dans la suite de ce travail de thèse. C'est ce type de modulation que nous avons exploité lors du développement du simulateur software, élaboré avec ST Microelectronics, dans le cadre du projet MEDEA+ A106 (INCA).

Dans le chapitre suivant nous allons préciser les aspects codages de la chaîne de transmission. Parmi les divers types de codage intervenant dans une chaîne de communication numérique, seul l'aspect codage canal sera abordé.

Chapitre 3

Turbo codes blocs

Sommaire

3.1 Introduction

Dans une chaîne de transmission numérique de l'information, le canal est souvent une source de perturbations. Pour lutter contre ces effets néfastes, il convient de mettre en place des systèmes de contre-mesures, permettant d'en atténuer les effets. Le canal de transmission est généralement bien modélisé par un filtre linéaire invariant dont la sortie

est bruitée par un bruit additif (voir Chapitre 2, figure 1). Ainsi, la première contre-mesure consiste naturellement à identifier la réponse impulsionnelle complexe du canal et à rechercher le filtre inverse permettant d'en minimiser les effets. Il est également possible, en plus de l'égaliseur, de donner au signal transmis des propriétés permettant de lutter contre les effets du canal et en particulier contre le bruit qu'il introduit, c'est le rôle du codage canal.

Ce chapitre sera donc consacré au codage canal, et en particulier aux codes correcteurs d'erreurs. Le codage canal consiste à introduire une redondance dans le signal d'information qui permettra au moins de détecter la présence d'erreurs dans le message reçu voire de corriger ces erreurs. De ce fait de nombreux codes canal existent et la sélection d'un codage canal particulier s'effectue à partir d'une connaissance "a priori" des effets du canal.

D'autres contraintes doivent également être prise en compte, telles que le débit binaire effectif souhaité et le taux d'erreur binaire admissible pour un bon fonctionnement du système. En effet, la redondance introduite par le codage canal permet d'améliorer le taux d'erreur dans un même temps qu'elle diminue le débit de transmission des symboles informatifs. Le choix d'un codage canal consiste donc à sélectionner des codes correcteurs permettant d'avoir des performances les plus proches possibles de la capacité théorique de Shannon au prix de la complexité la plus faible possible. Dans la pratique, la redondance est créée en ajoutant des symboles de contrôle qui s'inscrivent comme des combinaisons linéaires des symboles d'information.

Ce chapitre, aprés un bref historique sur le codage canal, commence par rappeler les principes de codage et de décodage des codes blocs algébriques classiques. Pour tirer pleinement parti des informations reçues à la sortie d'un canal de transmission, un décodage « souple » est nécessaire. Nous illustrerons quelques décodages souples mis en oeuvre pour les codes blocs puis nous exposerons les principes de l'algorithme de Pyndiah. Enfin, nous proposons une variante de l'algorithme de Pyndiah, plus simple à mettre en oeuvre grâce à une économie de mémoire importante. Cette dernière partie constitue une des contributions originales de ce travail de thèse. Elle a par ailleurs fait l'objet d'un dépot de brevet.

3.2 Généralités sur les codes blocs linéaires algébriques

3.2.1 Historique du codage canal

Nous avons vu précédemment que le canal de transmission est perturbé par du bruit aléatoire. Pour lutter contre cet effet néfaste du canal, on émet un signal qui appartient à un certain sous-ensemble de signaux possibles, ce sous-ensemble étant fixé à l'avance au niveau l'émetteur et connu du destinataire. Le but du codage canal est donc d'assurer une certaine fiabilité dans la transmission de l'information malgré les perturbations de la chaîne de transmission. Dans ce chapitre, nous allons examiner les principes de codes correcteurs puissants pouvant par exemple être proposés pour les systèmes hauts débits de la boucle locale filaire. Le codage canal consiste à définir des codes correcteurs permettant d'avoir des performances les plus proches possible de la capacité théorique

de Shannon du canal au prix d'une complexité la plus faible possible. Dans la pratique, la redondance créée pour y parvenir se fait à partir de l'ajout de symboles de contrôle qui s'écrivent comme des combinaisons linéaires des symboles d'information.

Ceci a donné naissance à deux grandes familles de codes : les codes convolutifs et les codes blocs polynomiaux. Les codes convolutifs sont constitués à partir de registres à décalage travaillant au fil de l'arrivée des symboles source ; ils fabriquent en sortie des symboles de redondance, qui sont des convolutions des informations avec la réponse impulsionnelle des registres à décalage rebouclés comme des filtres linéaires. Les codes blocs polynomiaux, au contraire, travaillent sur des blocs plus importants de données source, et considérés comme des polynômes dont les coefficients sont les symboles sources. Les symboles de redondance sont les coefficients du polynôme constituant le reste de la division du polynôme d'information par un polynôme fixé « générateur ». Comme nous allons le rappeler dans la suite de ce chapitre, le choix du polynôme générateur se fait par des considérations algébriques. On a parfois tendance à opposer ces deux familles de codes. Pour simplifier, disons que les codes convolutifs ont longtemps été adaptés à des canaux « plus difficiles », ayant une capacité plus faible, grâce à l'algorithme de Viterbi qui, depuis la fin des années 60, permet de travailler sur des décisions souples. Souvent en conséquence, on a utilisé pour des rendements inférieurs à $2/3$ les codes convolutifs, et les codes blocs pour des rendements plus importants dans des systèmes hauts débits. En fait à la fin du vingtième siècle, la plupart des systèmes utilisaient simultanément les qualités de ces deux types de codage avec la concaténation série classique d'un code algébrique de Reed Solomon externe avec un code convolutif interne. Mais même utilisé avec des constituants élémentaires puissants, ce schéma de codage butait à environ 2 dB de la limite de Shannon.

Il a fallu attendre la percée des turbo codes en 1993 pour qu'enfin, on puisse réaliser des schémas de codage approchant à moins de 1 dB de la limite de Shannon prédite un demi siècle plus tôt. Le schéma imaginé par Berrou et Glavieux repose sur le décodage réitéré plusieurs fois de l'algorithme MAP (Maximum A Posteriori) entre deux codes convolutifs. Cet algorithme est bien adapté aux codes ayant une structure en treillis régulière. En ce qui concerne les codes blocs algébriques, il a fallu attendre quelque temps pour qu'un algorithme turbo avec une complexité faible puisse être développé par R. Pyndiah.

3.2.2 Codes linéaires blocs

Pour les raisons évoquées précédemment, les codes blocs transforment k symboles d'un alphabet A, où par exemple $A = \mathbb{Z}/2\mathbb{Z}$, en des mots de codes de n symboles de l'alphabet avec $n > k$. Un code est par conséquent un sous ensemble des n-uplets possibles de l'alphabet et le récepteur cherche à retrouver le mot émis dans ce sous-ensemble. Le rendement $R = k/n$ quantifie la proportion de symboles réellement informatifs.

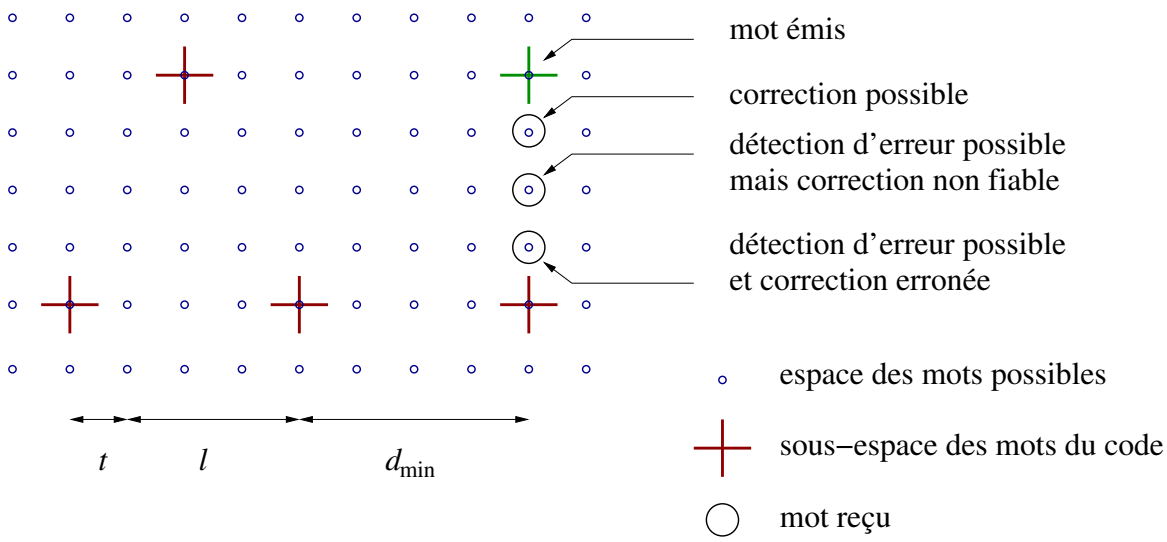


Figure 3.1: Représentation schématique de l'espace des mots de code.

Le nombre de symboles différents entre deux mots de codes constitue la « distance de Hamming » entre les deux mots ; pour les bruits uniformes vis-à-vis des symboles, plus cette distance est grande, et plus la probabilité qu'un mot codé se transforme, à cause des perturbations du canal de transmission, en l'autre mot codé est faible. Les performances d'un code seront notamment limitées par les mots du code les plus ressemblants, ou ayant une distance qui est la « distance minimale de Hamming », notée d_{min}. En particulier, la capacité de détection d'un code, à savoir le nombre de transitions maximum que l'on puisse à coup sûr détecter au niveau du récepteur, vaut $l = d_{min} - 1$; par ailleurs, si à la réception d'un n-uplet, un décodeur choisit parmi les mots du code, celui qui est à distance la plus faible du mot reçu, on peut garantir que la décision est juste tant que le nombre de transitions ne dépasse pas :

$$t = \left\lfloor \frac{d_{min} - 1}{2} \right\rfloor \tag{3.1}$$

où $\lfloor x \rfloor$ désigne le plus grand entier inférieur ou égal à x et l'entier t définit la "capacité de correction du code". Ceci peut être illustré par la figure 3.1.

Définition :

Un code linéaire C est un sous-espace vectoriel (s.e.v) de A^n engendré par k vecteurs indépendants (générateurs) $\mathbf{g}_1 \cdots \mathbf{g}_k$:

$$\mathbf{c} \in \mathbf{C} \Leftrightarrow \exists\,(a_i) \in A^k / \mathbf{c} = \sum_{i=1}^{k} a_i \mathbf{g}_i \Leftrightarrow C = Vect\,(\mathbf{g}_1, \cdots, \mathbf{g}_k) \tag{3.2}$$

Sous forme matricielle, si $\mathbf{g}_j = (g_{j,1} \cdots g_{j,n})$, (3.2) s'écrit aussi :

$$\mathbf{c} = \sum_{k=1}^{k} a_i \mathbf{g}_i = \left(\sum_{i=1}^{k} a_i g_{i,1}, \sum_{i=1}^{k} a_i g_{i,2}, \cdots, \sum_{i=1}^{k} a_i g_{i,n} \right) = \mathbf{aG} \tag{3.3}$$

où $\mathbf{a} = (a_1 \cdots a_k)$ et $\mathbf{G}$ matrice $k \times n$ dont les lignes sont formées par une famille de vecteurs générateurs $\mathbf{g}_i$ du code, est nommée « matrice génératrice » du code. La multiplication matricielle (3.3) traduit l'encodage du k-uplet $\mathbf{a}$ en le mot codé $\mathbf{c}$. En particulier, cet encodage est dit « sous forme systématique », lorsqu'on retrouve inchangées parmi les n coordonnées de $\mathbf{c}$, les k coordonnées d'informations originelles de $\mathbf{a}$.

Définition :
On appelle poids w d'un mot de code, le nombre de coordonnées non nulles du vecteur. Pour un code linéaire, l'ensemble des poids des mots du code coïncide avec l'ensemble des distances de Hamming entre mots du code car $d(\mathbf{c}_1, \mathbf{c}_2) = w(\mathbf{c}_1 - \mathbf{c}_2)$.

Définition :
Soit C un code linéaire, $C^\perp$ est un sous espace vectoriel de dimension $n-k$; c'est donc un code engendré par $n-k$ vecteurs notés $\mathbf{h}_1 \cdots \mathbf{h}_{n-k}$. Une matrice génératrice $\mathbf{H}$ de $C^\perp$ formée par $n-k$ lignes $\mathbf{h}_i$ est appelée matrice de contrôle de C.
Les lignes de $\mathbf{H}$ étant des mots de l'espace orthogonal à $\mathbf{C}$, on a :

$$\mathbf{c} \in \mathbf{C} \Leftrightarrow \mathbf{c}\mathbf{H}^T = 0 \tag{3.4}$$

La relation (3.4) est équivalente à la relation (3.3) ; ces relations décrivent les $n-k$ équations linéaires de l'hyperplan $\mathbf{C}$ de dimension k.

Définition :
On appelle syndrome d'un mot reçu, le vecteur à $n-k$ coordonnées de l'alphabet A, fabriqué par le décodeur du récepteur et défini par l'équation (3.5).

$$\mathbf{s} = \mathbf{r}\mathbf{H}^T \tag{3.5}$$

où $\mathbf{r} = \mathbf{c} + \mathbf{e}$ est le vecteur de n symboles reçus, $\mathbf{c}$ est le mot de code de n symboles émis et $\mathbf{e}$ est le vecteur des erreurs introduites par le canal entre la sortie du codeur de l'émetteur et l'entrée du décodeur du récepteur.

Application :
Si ce syndrome est non nul, on a détecté des erreurs de transmission car d'après (3.4), le mot reçu n'est pas un mot du code. De plus, le calcul du syndrome permet de distinguer à coup sûr toute configuration d'erreurs de poids inférieur à t, la capacité de correction du code ; en effet si $\mathbf{e}_i$ et $\mathbf{e}_j$ désignent deux configurations d'erreurs distinctes de poids inférieurs à t, les syndromes respectivement calculés sont différents.
En effet :

$$\mathbf{s}_i - \mathbf{s}_j = \mathbf{r}_i\mathbf{H}^T - \mathbf{r}_j\mathbf{H}^T = (\mathbf{c}_i + \mathbf{e}_i)\,\mathbf{H}^T - (\mathbf{c}_j + \mathbf{e}_j)\,\mathbf{H}^T = \mathbf{e}_i\mathbf{H}^T - \mathbf{e}_j\mathbf{H}^T = (\mathbf{e}_i - \mathbf{e}_j)\,\mathbf{H}^T \neq 0 \tag{3.6}$$

car $(\mathbf{e}_i - \mathbf{e}_j)$ de poids inférieur à d_{min} n'est pas un mot du code.
Une méthode de correction des erreurs peut donc être réalisée avec un tableau de correspondance consignant l'équivalence entre les $(Card\ A)^{n-k}$ valeurs possibles de syndromes et les configurations distinctes d'erreurs de poids correspondantes.
Un " décodage par tableau de correspondance " procède alors de la façon suivante :

- Le décodeur reçoit $\mathbf{r} = \mathbf{c} + \mathbf{e}$ et calcule le syndrome $\mathbf{s} = \mathbf{r}\mathbf{H}^T$.

- Il consulte le tableau de déchiffrement pour chercher la configuration d'erreur $\mathbf{e}_s$ correspondant à $\mathbf{s}$; nécessairement, si $w(\mathbf{e}) < t$, alors $\mathbf{e} = \mathbf{e}_s$.
- Il effectue la correction $\mathbf{c}' = \mathbf{r} - \mathbf{e}_s = \mathbf{c} + \mathbf{e} - \mathbf{e}_s = \mathbf{c}$ si $w(\mathbf{e}) < t$.

La méthode de décodage par tableau de correspondance permet de retrouver systématiquement le mot émis, tant que le canal ne produit pas plus d'erreurs que le code est capable de corriger. Sa complexité exponentielle avec la redondance $n-k$ est en général bien moindre qu'une recherche exhaustive par comparaison entre le mot reçu avec le sous-ensemble des mots possibles du code (maximum de vraisemblance de complexité $(Card\, A)^k$).

Propriété :
Il existe un mot du code de poids w si, et seulement si, il existe une combinaison linéaire de w colonnes de $\mathbf{H}$ qui soit elle-même nulle.

Démonstration :
Soit $\mathbf{c} = (c_1 \cdots c_n)$ un n-uplet ;
$\mathbf{c}$ a pour poids $w \Leftrightarrow Card\,\{c_i \neq 0\} = w$.
Or $\mathbf{c} \in C \Leftrightarrow \mathbf{H}\mathbf{c}^T = \mathbf{0}_{n-k}^T$

$$\Leftrightarrow \begin{pmatrix} h_{1,1} & \cdots & h_{1,n} \\ h_{2,1} & \cdots & h_{2,n} \\ \vdots & & \vdots \\ h_{n-k,1} & \cdots & h_{n-k,n} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \mathbf{0}$$

$$\Leftrightarrow \begin{cases} c_1 h_{1,1} + c_2 h_{1,2} + \cdots + c_n h_{1,n} = 0 \\ c_1 h_{2,1} + c_2 h_{2,2} + \cdots + c_n h_{2,n} = 0 \\ \vdots \\ c_1 h_{n-k,1} + c_2 h_{n-k,2} + \cdots + c_n h_{n-k,n} = 0 \end{cases}$$

$$\Leftrightarrow c_1 \begin{pmatrix} h_{1,1} \\ \vdots \\ h_{n-k,1} \end{pmatrix} + c_2 \begin{pmatrix} h_{1,2} \\ \vdots \\ h_{n-k,2} \end{pmatrix} + \cdots + c_n \begin{pmatrix} h_{1,n} \\ \vdots \\ h_{n-k,n} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

Donc le mot $\mathbf{c}$ du code a pour poids w si, et seulement si, il y a w coordonnées c_i qui sont non nulles et qu'il existe une combinaison linéaire de w colonnes de $\mathbf{H}$ qui est elle même nulle. CQFD.
Autrement dit, si $\mathbf{H}$ est une matrice telle que toute combinaison linéaire de $w-1$ ou moins colonnes est non nulle et qu'une combinaison linéaire de w colonnes est nulle alors :

$$d_{min} = w \tag{3.7}$$

3.2.3 Codes cycliques

Les codes cycliques sont une sous famille des codes linéaires pour lesquels par définition, toute permutation circulaire d'un mot de code est aussi un mot de code. Comme nous allons le voir ci-dessous, ceci permet une simplification matérielle au niveau du codeur, et du décodeur, en évitant les multiplications matricielles (3.3) et (3.4), compliquées à mettre en œuvre lorsque, comme pour la majorité des applications, la taille n du code devient grande.

Définition :
On appelle représentation polynomiale du vecteur $\mathbf{c} = (c_0 \cdots c_{n-1})$, le polynôme ayant les mêmes coefficients que $\mathbf{c}$, $C(X) = c_0 + c_1 X + \cdots + c_{n-1} X^{n-1}$.
Cette représentation polynomiale de degré $n-1$ peut-être vue comme le résultat d'une opération modulo un polynôme F de degré n et on choisit $F(X) = X^n - 1$. Nous allons caractériser $C[X]$, l'ensemble des représentations polynomiales des mots de code.

Propriété :
Un code C est cyclique si, et seulement si, tout multiple modulo $X^n - 1$ d'une représentation polynomiale de $C[X]$ est dans $C[X]$.

Démonstration :
A $\mathbf{c} = (c_0 \cdots c_{n-1})$ est associé $C(X) = c_0 + c_1 X + \cdots + c_{n-1} X^{n-1}$;
Soit $\mathbf{c}_p$une permutation circulaire vers la droite de $\mathbf{c}$:
A $\mathbf{c}_p = (c_{n-1} c_0 \cdots c_{n-2})$ est associé $C_p(X) = c_{n-1} + c_0 X + \cdots + c_{n-2} X^{n-1}$
$XC(X) = c_0 X + c_1 X^2 + \cdots + c_{n-1} X^n = c_{n-1} + c_0 X + \cdots + c_{n-2} X^{n-1}[X^n - 1] = C_p(X)$
donc $XC(X) \in C[X]$
De même pour tout entier i, $X^i C(X) \in C[X]$. Comme le code est linéaire, $(\sum_i p_i X^i) C(X)$, où les p_i sont des nombres binaires quelconques, est aussi un mot du code : $P(X) C(X) \in C[X]$.
Réciproquement, si pour tout entier i, $X^i C(X) \in C[X]$, toute permutation cyclique est bien un mot du code, C est un code cyclique. CQFD

Propriété :
Un code est cyclique si, et seulement si, toute représentation polynomiale est multiple d'un unique polynôme (de coefficient dominant 1) appelé polynôme générateur $g(X)$.

Démonstration :
Soit $g(X)$ la représentation polynomiale d'un mot du code de plus bas degré possible de coefficient dominant 1.
Soit $C(X) \in C[X]$. Effectuons la division de C par g :
$C = qg + r$ avec $d^o r < d^o g \Leftrightarrow C - qg = r$ avec C et qg éléments de C linéaire : $r \in C[X]$
Donc $r = 0$ car g est de degré minimum dans $C[X]$ et $C = qg$.
CQFD

Propriété :
Soit $A = G(p^r)$ un alphabet qui est un corps fini à p^r éléments, et n un entier premier avec p. Tout code cyclique de longueur n est engendré par un diviseur $g(X)$ de $X^n - 1$. Réciproquement, tout diviseur de $X^n - 1$ engendre un code cyclique.

Démonstration :
Effectuons la division de $X^n - 1$ par g:
$X^n - 1 = qg + r = 0\,[X^n - 1] \Leftrightarrow r = -qg\,[X^n - 1]$ donc
$\left.\begin{array}{l} r \in C[X] \\ d^o r < d^o g \end{array}\right\} \Rightarrow r = 0$: g est bien un diviseur de $X^n - 1$.
Réciproquement et par définition même d'un code cyclique, en considérant modulo $X^n - 1$ les multiples d'un polynôme g, en particulier diviseur de $X^n - 1$, on obtient bien un code cyclique.
CQFD

Exemple :
Soit $A = \mathbb{Z}/2\mathbb{Z}$, et $n = 7$; on a la factorisation : $X^7 - 1 = (X+1)(X^3 + X + 1)(X^3 + X^2 + 1)$
Le produit g des deux premiers facteurs s'écrit : $g(X) = X^4 + X^3 + X^2 + 1$
Ecrivons à présent tous les multiples de g de degré inférieur à 7 :

qg	$C(X)$	**C**
$0g(X)$	0	0000000
$1g(X)$	$1 + X^2 + X^3 + X^4$	1011100
$Xg(X)$	$X + X^3 + X^4 + X^5$	0101110
$(X+1)g(X)$	$1 + X + X^2 + X^5$	1110010
$X^2 g(X)$	$X^2 + X^4 + X^5 + X^6$	0010111
$(X^2+1)g(X)$	$1 + X^3 + X^5 + X^6$	1001011
$(X^2+X)g(X)$	$X + X^2 + X^3 + X^6$	0111001
$(X^2+X+1)g(X)$	$1 + X + X^4 + X^6$	1100101

On remarque que les mots associés aux représentations polynomiales sont bien des permutations circulaires les uns des autres.

Propriété :
Pour obtenir un codage sous forme systématique, il suffit, en opérant sur la représentation polynomiale du k-uplet à encoder, d'effectuer les trois opérations suivantes :

- Prémultiplier la représentation polynomiale $a(X)$ du message source à coder par X^{n-k}.
- Obtenir le reste $r(X)$ de la division de $X^{n-k}a(X)$par $g(X)$.

- Additionner $r(X)$ et $X^{n-k}a(X)$.

Démonstration :
Soit $\underline{a} = (a_0 \cdots a_{k-1})$ l'information source, $a(X) = a_0 + a_1 X + \cdots + a_{k-1}X^{k-1}$ est sa représentation polynomiale $X^{n-k}a(X) = a_0 X^{n-k} + a_1 X^{n-k+1} + \cdots + a_{k-1}X^{n-1}$ est un polynôme dont les degrés sont compris entre $n-1$ et $n-k$.
Divisons ce polynôme par $g(X)$ (de $d^o\, n-k$) ; le reste $r(X)$est de degré plus petit que $d^o g$:$d^o r < n-k$.
$\left(X^{n-k}a(X)\right) = q(X)\, g(X) + r(X)$ avec $r(X) = r_0 + \cdots + r_{n-k-1}X^{n-k-1}$
$\Leftrightarrow X^{n-k}a(X) - r(X) = q(X)\, g(X)$
$\Leftrightarrow X^{n-k}a(X) + r(X) = q(X)\, g(X)$(caractéristique $p = 2$)

$C(X) = X^{n-k}i(X) + r(X)$ étant multiple de $g(X)$, c'est la représentation polynomiale d'un mot du code polynomial. De plus $C(X) = r_0 + \cdots + r_{n-k-1}X^{n-k-1} + a_0 X^{n-k} + \cdots + a_{k-1}X^{n-1}$.
$C(X)$ est donc la représentation polynomiale de $\mathbf{c} = (r_0, \cdots, r_{n-k-1}, a_0, \cdots, a_{k-2}, a_{k-1})$ mot contenant les k informations source en tête.
CQFD

Un avantage immédiat des codes cycliques est donc que le codage sous forme systématique est réalisé avec un simple registre à décalage effectuant une division. Ainsi, diviser $b(X) = b_0 + b_1 X + \cdots + b_{\lambda-1}X^{\lambda-1}$ par $g(X) = g_0 + \cdots + g_{n-k-1}X^{n-k-1} + X^{n-k}$ s'effectue en $(\lambda+1)$ coups d'horloge par le registre (initialisé à 0), où les coefficients du dividende sont introduits à gauche du registre (fig. 3.2).

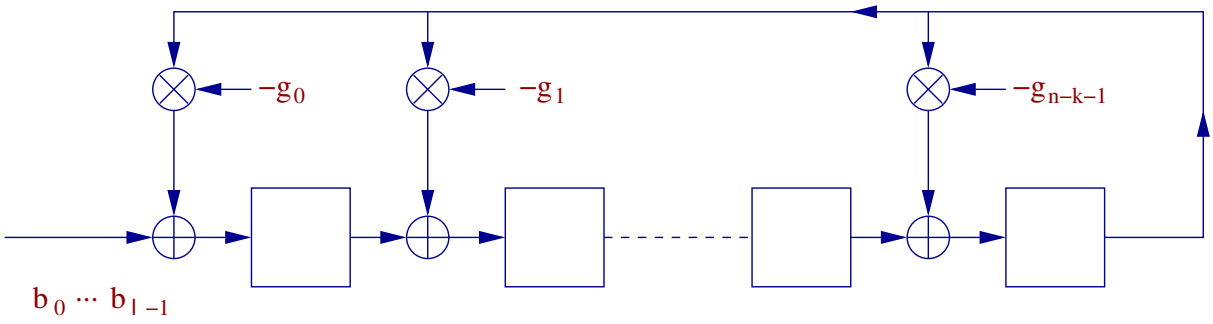


Figure 3.2: Division polynomiale par registre à décalage.

Dans le cas d'un polynôme à coefficients binaires, $g_i = 1$ (resp. $g_i = 0$) revient simplement à rétroactionner (resp. ne pas rétroactionner) le contenu de la dernière bascule à droite du registre.

3.2.4 Codes BCH

Les codes BCH [BRC60] tirent leur nom de leurs inventeurs : Bose, Chaudhuri, Hocquenghem. Ce sont les codes cycliques les plus utilisés car, d'une part, on peut garantir leur capacité de correction, et d'autre part, on peut réaliser des décodeurs avec une complexité tout à fait raisonnable.

Définition :
Soit α un élément générateur du corps $G\left(2^l\right)$. Au sens restreint du terme, un code BCH de longueur $n = 2^l - 1$ est un code dont le polynôme générateur contient plusieurs puissances consécutives de α.

Exemple :

$$g(X) = PPCM\left(f_1(X), f_2(X), \cdots, f_{2t}(X)\right) \tag{3.8}$$

où f_i est le polynôme minimal de α^i(Le fait de prendre le Plus Petit Commun Multiple permet de ne prendre qu'une fois en facteur dans $g(X)$ un polynôme minimal commun à différents éléments conjugués).

Propriété :
Les vecteurs lignes de

$$\mathbf{H}_0 = \begin{pmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & \cdots & \cdots & \left(\alpha^2\right)^{n-1} \\ \vdots & \vdots & & & \vdots \\ 1 & \alpha^{2t} & \cdots & \cdots & \left(\alpha^{2t}\right)^{n-1} \end{pmatrix}$$

sont dans $C^{\perp}$, l'orthogonal au code BCH engendré par le polynôme générateur $g(X)$.

Démonstration :
Soit $\mathbf{c}$ un mot du code dont la représentation polynomiale s'écrit : $C(X) = c_0 + c_1 X + \cdots + c_{n-1} X^{n-1}$
$C(X)$ est divisible par le polynôme générateur : $C(X) = q(X)\, g(X)$

$$\begin{aligned} &\Rightarrow \forall i \leq 2t \quad C\left(\alpha^i\right) = q\left(\alpha^i\right) g\left(\alpha^i\right) = q\left(\alpha^i\right) 0 = 0 \\ &\Rightarrow \forall i \leq 2t \quad c_0 + c_1 \alpha^i + \cdots + c_{n-1}\left(\alpha^i\right)^{n-1} = 0 \\ &\Rightarrow \forall i < 2t \quad (c_0 \cdots c_{n-1}) \begin{pmatrix} 1 \\ \vdots \\ \left(\alpha^i\right)^{n-1} \end{pmatrix} = 0 \end{aligned}$$

Les $2t$ lignes de $\mathbf{H}_0$ sont donc bien dans $C^{\perp}$ et $\mathbf{c}\mathbf{H}_0^T = \mathbf{0}$.
CQFD

"Le Théorème BCH"

Un code BCH dont le polynôme possède $2t$ racines consécutives (modulo n) permet de corriger t erreurs.

Démonstration :
On va montrer que $d_{min} > 2t+1 \Leftrightarrow 2t$ colonnes de $\mathbf{H}$ ou moins sont indépendantes nécessairement.
Supposons le contraire : il existe un mot de code $\mathbf{c} = (c_0 \cdots c_{n-1})$ ayant un poids ε inférieur ou égal à $2t$; on a donc :

$$\mathbf{0} = \mathbf{c}\mathbf{H}^T \Leftrightarrow \mathbf{0} = (c_0 \cdots c_{n-1}) \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ \alpha & \alpha^2 & \alpha^3 & \cdots & \alpha^{2t} \\ \alpha^2 & (\alpha^2)^2 & \cdots & \cdots & (\alpha^2)^{2t} \\ \vdots & \vdots & & & \vdots \\ \alpha^{n-1} & (\alpha^{n-1})^2 & \cdots & \cdots & (\alpha^{n-1})^{2t} \end{pmatrix}$$

Parmi les n composantes de $\mathbf{c}$, il y en a $n-\varepsilon$ qui sont nulles et n'apportent aucune contribution aux équations linéaires précédentes. Soient $c_{j_1} \cdots c_{j_\varepsilon}$ les composantes non nulles de $\mathbf{c}$:

$$\mathbf{0} = (c_{j1} c_{j2} \cdots c_{j\varepsilon}) \begin{pmatrix} \alpha^{j_1} & (\alpha^{j_1})^2 & \cdots & (\alpha^{j_1})^{2t} \\ \vdots & & & \vdots \\ \alpha^{j_\varepsilon} & \cdots & \cdots & (\alpha^{j_\varepsilon})^{2t} \end{pmatrix}$$

Si on pose $\alpha^{j_1} = X_1, \cdots, \alpha^{j_i} = X_i, \cdots, \alpha^{j_\varepsilon} = X_\varepsilon$, l'équation précédente devient :

$$\mathbf{0} = (c_{j_1} c_{j_2} \cdots c_{j_\varepsilon}) \begin{pmatrix} X_1 & X_1^2 & \cdots & X_1^{2t} \\ \vdots & & & \vdots \\ X_\varepsilon & \cdots & \cdots & X_\varepsilon^{2t} \end{pmatrix}$$

Cette matrice $\varepsilon \times 2t$ avec $\varepsilon \leq 2t$ a donc un noyau non nul : $rangM + dinKer = \varepsilon$

$$\Rightarrow rangM < \varepsilon < 2t$$
$$\Rightarrow Det = \begin{vmatrix} X_1 & X_1^2 & \cdots & X_1^{\varepsilon} \\ \vdots & & & \vdots \\ X_\varepsilon & \cdots & \cdots & X_\varepsilon^{\varepsilon} \end{vmatrix} = 0$$
$$\Rightarrow (X_1 X_2 \cdots X_\varepsilon) \begin{vmatrix} 1 & X_1 & \cdots & X_1^{\varepsilon-1} \\ \vdots & & & \vdots \\ 1 & \cdots & \cdots & X_\varepsilon^{\varepsilon-1} \end{vmatrix} = 0$$

Ceci est un déterminant "classique" de Van der Monde ; par récurrence sur sa dimension, on montre qu'il vaut $\prod_{m>n}(X_m - X_n) = \prod(\alpha^{j_m} - \alpha^{j_n}) \neq 0$: cette contradiction montre que l'hypothèse de trouver un mot de code $\mathbf{C}$ de poids plus petit que $2t$ est fausse.
CQFD

3.2.5 Mise en oeuvre du décodage d'un code BCH

Rappelons que le décodage d'un code n'est pas unique. Dans la pratique, on cherche à retrouver les mots les plus vraisemblablement émis avec la complexité la plus faible possible. Un décodage classique des codes BCH est le décodage algébrique. Il s'effectue en trois étapes :

- Calcul du syndrome sans multiplication matricielle.
- Passage du syndrome à la position des erreurs par le polynôme " localisateur des erreurs " et le système " d'équations-clés ".
- Obtention des racines du polynôme " localisateur des erreurs " donnant directement la position des erreurs.

3.2.5.1 Calcul du syndrome

$$\mathbf{s} = \mathbf{r}\mathbf{H}^T = (r_0 \cdots r_{n-1}) \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha & \alpha^2 & \cdots & (\alpha^{2t})^2 \\ \vdots & \vdots & & \vdots \\ \alpha^{n-1} & (\alpha^2)^{n-1} & \cdots & (\alpha^{2t})^{n-1} \end{pmatrix} \tag{3.9}$$

La $i^{ième}$ coordonnée du syndrome $\mathbf{s}$ s'écrit donc : $S_i = \sum_{j=0}^{n-1} r_j \alpha^{ij} = R(\alpha^i)$. Ceci est la représentation polynomiale du vecteur reçu évalué pour $X = \alpha^i$.
Or, si l'on fait la division de la représentation polynomiale $R(X)$ du mot reçu $\mathbf{r}$ par le polynôme minimal de α^i : $R(X) = q(X) f_i(X) + b^{(i)}(X)$ on a donc $R(\alpha^i) = b^{(i)}(\alpha^i)$.
L'évaluation du syndrome se fait donc en deux phases :

- Division de $R(X)$ par les polynômes minimaux facteurs de $g(X)$.
- Evaluation des restes pour $X = \alpha^i$

3.2.5.2 Passage du syndrome à la position des erreurs par le polynôme "localisateur des erreur"

A partir du calcul du syndrome, la correction se fait en deux temps :

- Construire un polynôme "localisateur des erreurs" à partir des valeurs du syndrome.
- Trouver les racines de ce polynôme "localisateur des erreurs", chaque racine correspondant à une position d'erreur.

On adopte les notations suivantes ; soit $\mathbf{c}$ le n-uplet émis et $C(X) = \sum_{i=0}^{n-1} c_i X^i$ sa représentation polynomiale. Soit $\mathbf{r}$ le n-uplet reçu ; $\mathbf{r}$ est entaché de ν erreurs aux emplacements $j_1, \cdots, j_\nu$ inconnus. $R(X)$, la représentation polynomiale de $\mathbf{r}$ est telle que :

$$R(X) = C(X) + E(X) \tag{3.10}$$

avec $E(X) = X^{j_1} + \cdots + X^{j_\nu}$, où $j_1 \cdots j_\nu$ sont les positions inconnues des erreurs et $\nu \leq t$ pour que la correction soit possible.
Calculer le syndrome revient à effectuer $2t$ calculs dans $G\left(2^\lambda\right)$: $S_i = R(\alpha^i)$ pour $1 \leq i \leq 2t$.
Or (3.10) $\Rightarrow R(\alpha^i) = C(\alpha^i) + E(\alpha^i) = E(\alpha^i)$ (car $C(\alpha^i) = 0$ puisque $C(X)$ est multiple de $g(X)$ qui a pour racine α^i). Donc les $2t$ coordonnées du syndrome sont liées aux emplacements des erreurs $j_1, \cdots, j_\nu$ par :

$$\begin{aligned} S_1 &= (\alpha)^{j_1} + (\alpha)^{j_2} + \cdots + (\alpha)^{j_\nu} \\ S_2 &= \left(\alpha^2\right)^{j_1} + \left(\alpha^2\right)^{j_2} + \cdots + \left(\alpha^2\right)^{j_\nu} \\ &\vdots \\ S_{2t} &= \left(\alpha^{2t}\right)^{j_1} + \left(\alpha^{2t}\right)^{j_2} + \cdots + \left(\alpha^{2t}\right)^{j_\nu} \end{aligned} \tag{3.11}$$

Pour simplifier les notations, on pose $\beta_i = \alpha^{j_i} (i \leq \nu)$.
Le système devient :

$$\begin{aligned} S_1 &= \beta_1 + \beta_2 + \cdots + \beta_\nu \\ S_2 &= \beta_1^2 + \beta_2^2 + \cdots + \beta_\nu^2 \\ &\vdots \\ S_{2t} &= \beta_1^{2t} + \beta_2^{2t} + \cdots + \beta_\nu^{2t} \end{aligned} \tag{3.12}$$

On obtient un système de $2t$ équations à de ν inconnues dans $G\left(2^l\right)$: les β_i.
Plutôt que de résoudre "en force" ce système, on va construire un polynôme qui permet d'en trouver automatiquement la solution.
Soit :

$$\begin{aligned} \sigma(X) &= (X - \beta_1)(X - \beta_2) \cdots (X - \beta_\nu) = (X + \beta_1)(X + \beta_2) \cdots (X + \beta_\nu) \\ &= X^\nu + \sigma_1 X^{\nu-1} + \cdots + \sigma_{\nu-1} X + \sigma_\nu \end{aligned} \tag{3.13}$$

On appelle $\sigma(X)$ le polynôme "localisateur des erreurs" en raison du fait que :

$$\forall i \leq \nu \leq t, \quad \sigma(\beta_i) = 0 \tag{3.14}$$

Entre les coefficients du polynôme et ses racines β_i, il y a les identités de Newton :

$$\sigma_1 = \sum_i \beta_i, \cdots, \sigma_j = \sum_{i_1 > \cdots > i_j} \beta_{i_1} \beta_{i_2} \cdots \beta_{i_j}, \cdots, \sigma_\nu = \beta_1 \beta_2 \cdots \beta_\nu \tag{3.15}$$

On voit que le premier coefficient $\sigma_1 = S_1$ et on cherche à exprimer les ν coefficients σ_i de ce polynôme en fonction des coordonnées du syndrome. Pour cela, on peut remarquer que :

$\sigma(X) = X^\nu + \sigma_1 X^{\nu-1} + \cdots + \sigma_{\nu-1} X + \sigma_\nu$
$\Rightarrow X^j \sigma(X) = X^{\nu+j} + \sigma_1 X^{j+\nu-1} + \cdots + \sigma_{\nu-1} X^{j+1} + \sigma_\nu X^j$ avec $j = 1, 2, \cdots, \nu$

$\Rightarrow$ pour $X = \beta_i$, $\beta_i^j \sigma(\beta_i) = \beta_i^{\nu+j} + \sigma_1 \beta_i^{\nu+j-1} \cdots + \sigma_{\nu-1} \beta_i^{j+1} + \sigma_v \beta_i^j$ avec $j = 1, 2, \cdots, \nu$

Mais comme par choix du polynôme localisation d'erreurs, $\sigma(\beta_i)$=0 pour tout i $\leq \nu$, on a :

$\sum_{i=1}^{\nu} \beta_i \sigma(\beta_i) = 0$ avec $j = 1, 2 \cdots \nu$
$\Leftrightarrow \left(\sum_{i=1}^{\nu} \beta_i^{\nu+j}\right) + \sigma_1 \left(\sum_{i=1}^{\nu} \beta_i^{\nu+j-1}\right) + \cdots + \sigma_\nu \left(\sum_{i=1}^{\nu} \beta_i^j\right) = 0$ avec $j = 1, 2 \cdots \nu$
$\Leftrightarrow S_{j+\nu} + \sigma_1 S_{j+\nu-1} + \cdots + \sigma_{\nu-1} S_{j+1} + \sigma_\nu S_j = 0$ avec $j = 1, 2 \cdots \nu$

donc :

$$\begin{aligned} &S_{\nu+1} + \sigma_1 S_\nu + \cdots + \sigma_{\nu-1} S_2 + \sigma_\nu S_1 = 0 \\ &S_{\nu+2} + \sigma_1 S_{\nu+1} + \cdots + \sigma_{\nu-1} S_3 + \sigma_\nu S_2 = 0 \\ &\vdots \\ &S_{2\nu} + \sigma_1 S_{2\nu-1} + \cdots + \sigma_{\nu-1} S_{\nu+1} + \sigma_\nu S_\nu = 0 \end{aligned} \tag{3.16}$$

Ceci est un système dit " équations-clés " de ν équations linéaires à ν inconnues : les σ_i (les coordonnées S_j du syndrome sont obtenues dans le récepteur à l'étape 3.2.5.1 précédente lors du calcul du syndrome).On tombe donc sur un système "classique" que l'on sait résoudre par différentes méthodes (cf paragraphe 3.2.5.4).

3.2.5.3 Racines du polynôme

Ayant un polynôme dont les racines $\beta_i = \alpha^{j_i}$ sont des éléments d'un corps fini $G\left(2^l\right) = G(n+1)$, il suffit d'essayer de tester un à un les n éléments non nuls du corps, pour conserver ceux qui sont racines de $\sigma(X)$. Cette réalisation peut être informatisée ou implantée électroniquement dans le cas d'une application ayant des contraintes de temps réel avec un décodeur type Chien [LC83].

3.2.5.4 Résolution du système d'équations-clés

Le calcul du polynôme " localisateur des erreurs " est l'étape critique du décodage en terme du nombre de calculs nécessaires. On a vu au paragraphe 3.2.5.2 que ce calcul revient à la résolution du système d'équations-clés (3.16) qui s'écrit aussi sous forme matricielle :

$$\begin{pmatrix} S_1 & S_2 & \cdots & S_{\nu-1} & S_\nu \\ S_2 & S_3 & \cdots & S_\nu & S_{\nu+1} \\ \vdots & \vdots & & \vdots & \vdots \\ S_\nu & S_{\nu+1} & \cdots & S_{2\nu-2} & S_{2\nu-1} \end{pmatrix} \begin{pmatrix} \sigma_\nu \\ \sigma_{\nu-1} \\ \vdots \\ \sigma_1 \end{pmatrix} = \begin{pmatrix} S_{\nu+1} \\ S_{\nu+2} \\ \vdots \\ S_{2\nu} \end{pmatrix} \tag{3.17}$$

où $v \leq t$ pour que la capacité de correction du code ne soit pas dépassée.

Les coefficients du polynôme localisateur d'erreur peuvent donc s'obtenir si la matrice carrée précédente $\mathbf{S}_\nu$ constituée par les syndromes est inversible. Un problème non abordé

jusqu'à présent est que le nombre d'erreurs ν est inconnu a priori du récepteur. Peterson [PW72] a montré que la matrice carrée $\mathbf{S}_\mu$constituée de syndromes,

$$\mathbf{S}_\mu = \begin{pmatrix} S_1 & S_2 & \cdots & S_{\mu-1} & S_\mu \\ S_2 & S_3 & \cdots & S_\mu & S_{\mu+1} \\ \vdots & \vdots & & \vdots & \vdots \\ S_\mu & S_{\mu+1} & \cdots & S_{2\mu-2} & S_{2\mu-1} \end{pmatrix} \tag{3.18}$$

est non inversible si $\mu > \nu$ le nombre d'erreurs introduites par le canal (car les syndromes ne dépendent en tout que de ν termes indépendants). Pour calculer le polynôme localisateur d'erreurs et inverser la matrice des syndromes, le récepteur peut donc procéder suivant l'algorithme de Peterson-Gorenstein-Zierler :

1/ Initialiser $\mu = t$ (nombre maximum d'erreurs que l'on peut garantir corriger)
2/ Calculer $det\,[\mathbf{S}_\mu]$
3/ Si $det\,[\mathbf{S}_\mu] \neq 0$ alors

$$\begin{pmatrix} \sigma_\mu \\ \sigma_{\mu-1} \\ \vdots \\ \sigma_1 \end{pmatrix} = \begin{pmatrix} S_{\mu+1} \\ S_{\mu+2} \\ \vdots \\ S_{2\mu} \end{pmatrix} \begin{pmatrix} S_1 & S_2 & \cdots & S_{\mu-1} & S_\mu \\ S_2 & S_3 & \cdots & S_\mu & S_{\mu+1} \\ \vdots & \vdots & & \vdots & \vdots \\ S_\mu & S_{\mu+1} & \cdots & S_{2\mu-2} & S_{2\mu-1} \end{pmatrix}^{-1}$$

4/ Sinon décrémenter d'une unité μ (nombre estimé d'erreurs introduites par le canal) et retourner à l'étape 2/

Exemples :

- Pour un code BCH 1-correcteur, le système d'équations clés se résume à tester si le syndrome est nul ou pas. Si le syndrome est non nul, sa valeur donne directement la position de l'erreur.
- Pour un code BCH 2-correcteur, (3.17) devient :

$$\begin{pmatrix} S_1 & S_2 \\ S_2 & S_3 \end{pmatrix} \begin{pmatrix} \sigma_2 \\ \sigma_1 \end{pmatrix} = \begin{pmatrix} S_3 \\ S_4 \end{pmatrix}$$

Si le déterminant $S_1S_3 - S_2^2 = 0$ est non nul, on obtient :

$$\begin{pmatrix} \sigma_2 \\ \sigma_1 \end{pmatrix} = \begin{pmatrix} S_1 & S_2 \\ S_2 & S_3 \end{pmatrix}^{-1} \begin{pmatrix} S_3 \\ S_4 \end{pmatrix} = \frac{1}{S_1S_3 - S_2^2} \begin{pmatrix} S_3 & -S_2 \\ -S_2 & S_1 \end{pmatrix} \begin{pmatrix} S_3 \\ S_4 \end{pmatrix}$$

soit $\sigma_1 = \frac{-S_2S_3+S_1S_4}{S_1S_3-S_2^2} = S_1$ (car $S_4 = S_2^2 = S_1^4$) et $\sigma_2 = \frac{S_3^2-S_2S_4}{S_1S_3-S_2^2} = \frac{S_3^2+\left(S_1^3\right)^2}{S_1\left(S_3+S_1^3\right)} = \frac{S_3}{S_1} + S_1^2$

Sinon si le déterminant est nul, on retombe sur l'équation-clé du code 1-correcteur.

Soit le corps de Galois $G(16)$ construit à partir des opérations modulo X^4+X+1 et considérons le code BCH (15,5) 3-correcteur engendré par :
$g(X) = f_1(X) f_3(X) f_5(X) = \left(X^4+X+1\right)\left(X^4+X^3+X^2+X+1\right)\left(X^2+X+1\right)$
Supposons que l'on reçoive $r(X) = X^2+X^7$.
Le récepteur commence par calculer les coordonnées du syndrome.
Il trouve : $S_1 = \alpha^{12}$, $S_2 = \alpha^9$, $S_3 = 0$, $S_4 = \alpha^3$, $S_5 = 1$, $S_6 = 0$.
D'après Peterson, Gorenstein et Zierler, on pose μ=3 et on vérifie si le déterminant de $\mathbf{S}_3$ est nul ou non pour savoir s'il y a eu 3 erreurs, nombre maximum que ce code garantit de corriger.

$\mu = 3$ et $det[\mathbf{S}_3] = \begin{vmatrix} S_1 & S_2 & S_3 \\ S_2 & S_3 & S_4 \\ S_3 & S_4 & S_5 \end{vmatrix} = \begin{vmatrix} \alpha^{12} & \alpha^9 & 0 \\ \alpha^9 & 0 & \alpha^3 \\ 0 & \alpha^3 & 1 \end{vmatrix} = 0$: il n'y a pas eu trois erreurs ; on décrémente donc μ :

$\mu = 2$ et $det[\mathbf{S}_2] = \begin{vmatrix} S_1 & S_2 \\ S_2 & S_3 \end{vmatrix} = \begin{vmatrix} \alpha^{12} & \alpha^9 \\ \alpha^9 & 0 \end{vmatrix} = \alpha^3$ Le récepteur considère donc qu'il y a eu deux erreurs. D'où :

$$\begin{aligned} \begin{pmatrix} \sigma_2 \\ \sigma_1 \end{pmatrix} &= \begin{pmatrix} S_1 & S_2 \\ S_2 & S_3 \end{pmatrix}^{-1} \begin{pmatrix} S_3 \\ S_4 \end{pmatrix} \\ &= \frac{1}{S_1 S_3 - S_2^2} \begin{pmatrix} S_3 & -S_2 \\ -S_2 & S_1 \end{pmatrix} \begin{pmatrix} S_3 \\ S_4 \end{pmatrix} \\ &= \begin{pmatrix} 0 & \alpha^6 \\ \alpha^6 & \alpha^9 \end{pmatrix} \begin{pmatrix} 0 \\ \alpha^3 \end{pmatrix} \\ &= \begin{pmatrix} \alpha^9 \\ \alpha^{12} \end{pmatrix} \end{aligned}$$

Ceci permet de former le polynôme localisateur : $X^2+\alpha^{12}X+\alpha^9 = 0$ dont les racines sont α^2 et α^7. Le récepteur obtient donc les emplacements de deux erreurs et corrige le mot reçu en trouvant pour mot émis, le mot de code nul.

L'algorithme de Peterson-Gorenstein-Zierler n'a pas la complexité exponentielle d'une méthode type tableau de déchiffrement. Il repose sur l'inversion de matrices, et reste commode tant que la taille des matrices ($\mu \times \mu$) n'est pas importante (complexité de calcul polynomiale en μ^3). On utilise ce procédé pour tous les codes algébriques de capacité de correction plus petit que $t = 5$. Lorsque l'on désire utiliser des codes de capacité de correction plus importante, il existe des méthodes pour obtenir le polynôme localisateur, théoriquement plus élaborées que la simple inversion de matrices, mais qui sont moins complexes à mettre en oeuvre. En particulier, la méthode de Berlekamp-Massey [Ber82][Mas69] se base sur la symétrie des coefficients du système linéaire (3.17) pour obtenir les coefficients de $\sigma(X)$ de façon itérative. En effet, le système (3.17) est un ensemble d'équations de convolutions entre la suite connue des coefficients du syndrome et la suite inconnue des coefficients du polynôme localisateur d'erreurs. L'algorithme de Berlekamp-Massey consiste à synthétiser le registre à décalage contenant la suite des

syndromes, de manière à ce que les pondérations de chaque sortie du registre vérifient itérativement les équations (3.17). Sa complexité calculatoire est aussi polynomiale mais simplement en $\mathcal{O}(6t^2)$. Il a été montré [CDG92] que cet algorithme constitue une variation de l'algorithme d'Euclide. D'autres méthodes existent mais d'excellents turbo codes sont constitués à partir de codes élémentaires simples avec des faibles capacités de correction. L'algorithme de Peterson-Gorenstein-Zierler est donc le plus approprié en ce qui nous concerne.

3.2.6 Codes RS

Les codes RS [RS60] tirent leur nom de leurs inventeurs Irving Reed et Gus Solomon au début des années 60. Plutôt que de transmettre des éléments binaires, on envisage l'émission de symboles dans $G(2^m)$. Ainsi, transmettre γ de $G(2^m)$ revient à transmettre les m coordonnées binaires de sa représentation polynomiale. On considère alors les codes cycliques de longueur $n = 2^m - 1$ symboles de $G(2^m)$; par exemple pour $m = 4$, on prendra des codes de 15 symboles caractérisés par 4 bits, soit 60 bits. Les racines $n^{ièmes}$ de l'unité se trouvent dans $G\left(2^l\right)$ où l est le plus petit entier tel que n divise $2^l - 1$: il s'agit donc du même corps $G(2^m) = G(n+1)$.

Un code de Reed Solomon est un code type BCH, mais à coefficients non binaires, dont le polynôme générateur possède $2t$ racines qui sont des puissances consécutives dans $G(2^m)$. Souvent on choisit :

$$g(X) = (X - \alpha)\left(X - \alpha^2\right) \cdots \left(X - \alpha^{2t}\right) = PPCM_{G(2^m)}\left(f_1(X)\, f_2(X) \cdots f_{2t}(X)\right) \tag{3.19}$$

Comme pour les codes BCH, mais en considérant ici des symboles non binaires, on en déduit les caractéristiques suivantes (exprimées en symboles) :

$n = 2^m - 1$

$n - k = d^o g = 2t$

$d_{min} = 2t + 1$, capacité de correction $e = t$ symboles.

Le codage se fait classiquement par une division polynomiale ; la seule nuance par rapport aux codes BCH présentés au paragraphe précédent est que le polynôme générateur (3.19) n'a aucune raison d'être à coefficients binaires puisque de façon générale, il ne contient pas comme racine des classes d'éléments conjugués.

Lors d'une transmission, la perte de $m + 1$ bits consécutifs n'affecte pas plus de deux symboles (m bits par symbole). De même la perte de $(2m + 1)$ bits consécutifs n'affecte pas plus de 3 symboles et d'une manière plus générale, la perte d'un paquet de $(t - 1)\, m + 1$ bits n'affecte pas plus de t symboles.

Un code RS de capacité de correction t symboles peut donc :

- Corriger t erreurs binaires aléatoires parmi tm bits (avec un meilleur rendement qu'un code BCH binaire car le code de Reed Solomon atteint la borne de Singleton).

- Corriger un paquet de $[(t-1)m+1]$ bits erronés (affectant au plus t symboles consécutifs).

De manière analogue aux codes BCH binaires, le décodage s'effectue par une méthode algébrique se déroulant en plusieurs étapes :

- Calcul du syndrome
- Emploi d'un polynôme " localisateur d'erreurs "
- Correction des erreurs ; la principale différence réside en ce point car les symboles n'étant pas binaires, il faut outre la localisation des erreurs, retrouver quels symboles de m bits avaient été émis aux positions incriminées.

3.2.6.1 Calcul du syndrome

Comme pour les codes BCH, les $2t$ composantes du syndrome s'obtiennent en effectuant la division de la représentation polynomiale du mot reçu $R(X)$ par les $2t$ facteurs du polynôme générateur g (cf (3.19)) et en l'évaluant en $X=\alpha^i$: $S_i = R(\alpha^i)$. Ceci est donc réalisé pour tout entier i plus petit que $2t$, par la division $R(X) = q(X)(X-\alpha^i) + b^{(i)}(X)$ où $d^o b^{(i)} < 1$; $b^{(i)}(X)$ est donc une constante de $G(2^m)$, obtenue par le registre effectuant la division par $(X-\alpha^i) = X+\alpha^i$ $(p=2)$, en n coups d'horloge (fig. 3.3).

3.2.6.2 Mise en équation et correction à l'aide du polynôme "localisateur des erreurs"

Soit $e(X)$ le "polynôme erreur" qui, comme le code émis est une suite de symboles dans $G(2^m)$:

$e(X) = Y_1 X^{e_1} + Y_2 X^{e_2} + \cdots + Y_\nu X^{e_\nu}$ où e_i est l'emplacement du $i^{ième}$ symbole erroné et Y_i de $G(2^m)$ est l'erreur s'étant produite sur ce groupe de m bits.

De manière analogue aux codes BCH binaires, les premières coordonnées du syndrôme s'écrivent :

$S_1 = R(X=\alpha^1) = C(X=\alpha^1) + e(X=\alpha^1) = e(X=\alpha^1)$

$S_1 = Y_1\alpha^{j_1} + \cdots + Y_\nu\alpha^{j_\nu} = \sum_{1\le i\le\nu} Y_i\beta_i$

en posant $\beta_i = \alpha^{j_i}$,

$S_2 = R(X=\alpha^2) = Y_1(\alpha^2)^{j_1} + \cdots + Y_\nu(\alpha^2)^{j_\nu} = \sum_{1\le i\le\nu} Y_i\beta_i^2$

On obtient donc ainsi pour les $2t$ coordonnées du syndrome le système d'équation (3.20).

$$\begin{aligned} S_1 &= R\left(X=\alpha^1\right) = e\left(X=\alpha^1\right) = \sum_{1\le i\le\nu} Y_i\beta_i \\ &\vdots \\ S_l &= R\left(X=\alpha^l\right) = e\left(X=\alpha^l\right) = Y_1\left(\alpha^{j_1}\right)^l + \cdots + Y_\nu\left(\alpha^{j_\nu}\right)^l = \sum_{1\le i\le\nu} Y_i(\beta_i)^l \quad (3.20) \\ &\vdots \end{aligned}$$

$$\begin{aligned} S_{2\nu} &= R\left(X=\alpha^{2\nu}\right) = e\left(X=\alpha^{2\nu}\right) = Y_1\left(\alpha^{j_1}\right)^{2\nu} + \cdots + Y_\nu\left(\alpha^{j_\nu}\right)^{2\nu} = \sum_{1\le i\le\nu} Y_i\left(\beta_i\right)^{2\nu} \\ &\vdots \\ S_{2t} &= R\left(X=\alpha^{2t}\right) = \sum_{1\le i\le\nu} Y_i\left(\beta_i\right)^{2t} \end{aligned}$$

Les coordonnées du syndrome S_i étant calculées à l'étape précédente, (3.20) est un système de $2t$ équations à 2ν inconnues : les β_i et Y_i, éléments de $G\left(2^m\right)$. Lorsque $t<\nu$, la capacité de correction du code est dépassée et l'on ne peut garantir un résultat que pour $\nu \le t$. Comme pour les codes BCH binaires, plutôt que de résoudre en force un tel système, Gorenstein et Zierler ont proposés de fabriquer un polynôme localisation des erreurs :

$$\sigma(X) = (X+\beta_i)\cdots(X+\beta_\nu) = X^\nu + \sigma_1 X^{\nu-1} + \cdots + \sigma_\nu \tag{3.21}$$

qui débouche sur exactement les mêmes identités de Newton et donc le même système d'équations linéaires (3.16). Ce système linéaire est donc résolu par les mêmes algorithmes que dans le cas des codes BCH binaires, par exemple l'algorithme de Peterson-Gorenstein-Zierler. La connaissance des β_i induit que (3.20) devient simplement lui même alors, un système d'équations linéaires d'inconnues Y_i que l'on sait tout autant résoudre par une méthode classique (pivot de Gauss, [For65],...).

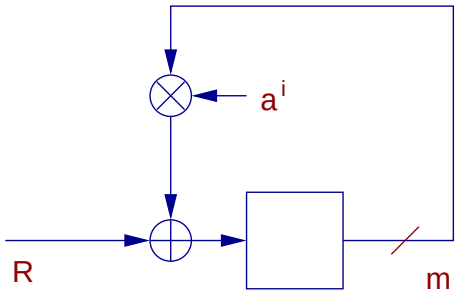


Figure 3.3: Calcul du syndrôme d'un code de Reed-Solomon par division par registres à décalage.

Ce paragraphe avait pour objet de résumer les propriétés les plus importantes des codes algébriques sur lesquels les turbo codes blocs se basent. Pour plus de détails, on pourra consulter les références bibliographiques [PW72][CDG92][Gel01][Hoc59].

3.3 Décodage souple

La construction des codes algébriques à symboles dans des corps de caractéristique 2 a naturellement généré dans un premier temps les méthodes de décodage "dures" décrites ci-dessus, où le décodeur ne traite que des nombres binaires. Toutefois, lorsque le récepteur prend de telles décisions binaires en amont du décodeur, il efface une partie de l'historique

que le canal de transmission a imposé aux symboles le traversant ; avec de telles décisions "dures" -ou hard-, le décodeur ignore la fiabilité de chacun des symboles reçus ; par exemple, dans le cas d'une modulation binaire, la sortie du détecteur en amont de la chaîne de réception numérique est elle conforme à l'énergie à laquelle on s'attend, ou au contraire est-elle proche du seuil de décision entre les deux valeurs possibles ? En d'autres termes, si le signal reçu est proche en terme de distance euclidienne dans le plan complexe des symboles possibles de la QAM, on peut estimer que la décision "dure" est fiable ; au contraire, si le signal reçu est distant d'un symbole QAM, la décision "dure" est plus hasardeuse et moins fiable, et pourtant elle pèse autant au niveau d'un mot de code, qu'une décision fiable. Ce problème est bien connu des traiteurs de l'information [Bat98][CT91][Sha48], et on peut montrer qu'à conditions équivalentes de transmission, la capacité du canal à sortie souple est plus grande que celle du canal à sortie dure. Cette capacité ne peut être approchée qu'au prix d'un traitement plus compliqué de l'information, manipulant notamment des nombres qui ne sont plus discrets.

3.3.1 Décodeur à effacements

Un premier pas vers le décodage souple consiste à considérer que si l'alphabet d'origine peut prendre deux valeurs binaires possibles $\{0_e, 1_e\}$, par contre on distingue à l'entrée du décodeur, trois valeurs $\{0_s, 1_s, \text{Eff}\}$, où Eff désigne un effacement du symbole lorsque le signal reçu est trop éloigné des valeurs considérées comme fiables correspondants au 0_s ou au 1_s.

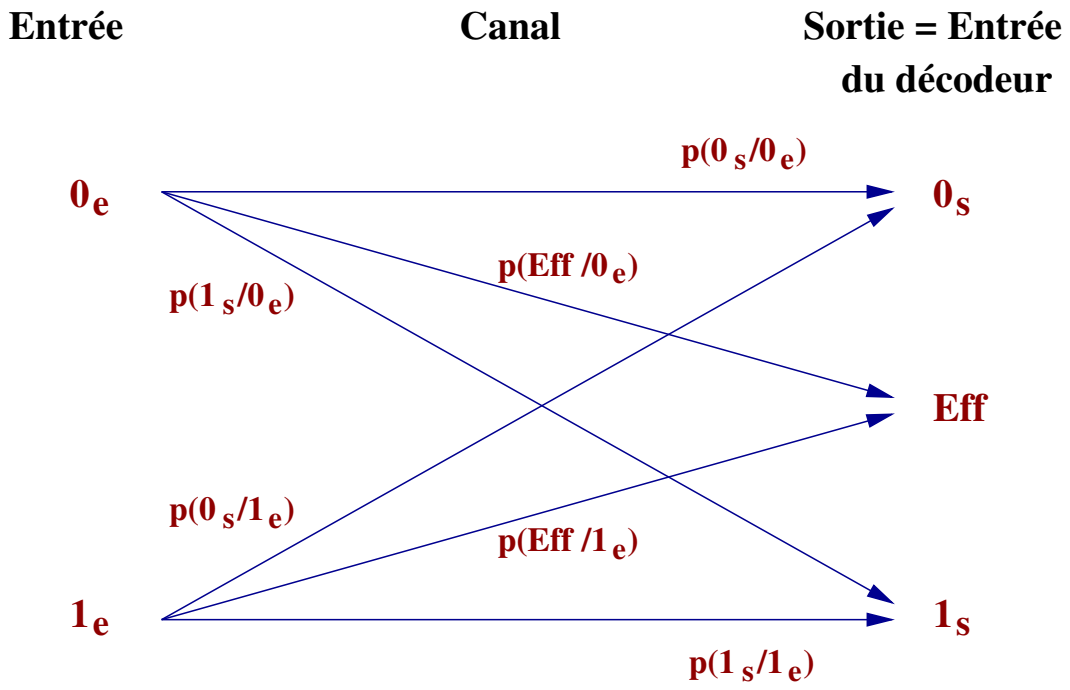


Figure 3.4: Canal à entrée binaire et à sortie ternaire.

Un code bloc linéaire de distance minimale d_{min} est aussi capable de simultanément corriger t_e erreurs et f effacements du moment que :

$$2t_e + f < d_{min} \tag{3.22}$$

En effet, si l'on ne considère dans les différents mots du code que les coordonnées non effacées, la distance minimale entre les mots " raccourcis " résultants est au moins de $d_{min} - f$, d'où une capacité de correction (cf (3.1)) de $t = \left\lfloor \frac{(d_{min}-f)-1}{2} \right\rfloor$, ce qui est bien équivalent à (3.22). Cette formule traduit donc le fait qu'un code est capable de corriger deux fois plus d'effacements que d'erreurs. Nous allons à présent montrer que cette amélioration est possible pour les codes BCH au prix d'une augmentation raisonnable de complexité, en adaptant les décodeurs du paragraphe 3.2.5.

Décodage avec effacement :

Soit $\mathbf{r}_0$ (resp. $\mathbf{r}_1$) le mot obtenu en remplaçant dans le mot reçu $\mathbf{r}$ tous les symboles effacés jugés peu fiables par des 0 (resp. des 1). En appliquant successivement à l'entrée du décodeur algébrique classique, les deux mots $\mathbf{r}_0$ et $\mathbf{r}_1$, le décodeur fournit deux vecteurs $\mathbf{c}_0$ et $\mathbf{c}_1$(pas forcement distincts). Le décodage consiste alors à garder parmi ces deux mots celui qui est à distance minimale du mot initialement reçu $\mathbf{r}$.

Propriété :

La méthode de décodage par effacement précédente, permet d'obtenir le mot le plus proche au sens de la distance de Hamming, tant que la relation (3.22) est vérifiée.

Démonstration :

Supposons que le mot reçu $\mathbf{r}$ est entaché d'au plus t_e erreurs.

En mettant les e_f symboles effacés à 0, on rajoute e_0 erreurs dans $\mathbf{r}_0$. De même, en mettant les symboles effacés à 1, on rajoute $e_1 = e_f - e_0$ erreurs dans $\mathbf{r}_1$. Suivant si plus ou moins de la moitié des bits effacés valaient 0 ou 1, au moins l'un des deux nombres e_0 ou $e_f - e_0$ est plus petit que $e_f/2$. Donc si e_t désigne $min\,(t_e + e_0, t_e + e_1)$, à savoir le plus petit nombre d'erreurs parmi $\mathbf{r}_0$ et $\mathbf{r}_1$, on a par conséquent :

$2e_t \leq 2\left(t_e + \frac{e_f}{2}\right) = 2t_e + e_f < d_{min}$

Par conséquent, l'un des deux mots $\mathbf{r}_0$ ou $\mathbf{r}_1$est bien à une distance $e_t < d_{min}/2$ du mot de code original et le décodage algébrique fournira la décision correcte.

CQFD.

Le décodage par effacement et correction augmente donc finalement le nombre total de symboles que l'on peut corriger. Les codes algébriques classiques conviennent à ce type de décodage souple. Seul le décodage est légèrement modifié, et si l'on peut finalement corriger "deux fois plus d'effacements que d'erreurs", c'est parce que l'on facilite la connaissance a priori du décodeur en lui fournissant déjà les emplacements susceptibles d'être entachés d'erreurs.

3.3.2 Décodage de Chase

Nous allons à présent présenter une classe d'algorithmes de décodage [Cha72] capables de réaliser un décodage à entrée souple à partir de n'importe quel décodeur bloc algébrique à entrée dure. Ces algorithmes, souvent nommés Chasing, permettent d'obtenir pratiquement un facteur deux sur la capacité de correction par rapport à la relation (3.1) du décodage dur (en l'occurrence, jusqu'à $d_{min} - 1$ erreurs).

3.3.2.1 Présentation du problème

Reprenons les notations de Chase. Chaque séquence d'information de longueur k est supposée codée en un bloc de n bits noté :

$$\mathbf{X} = (X_1, X_2, ..., X_n) \tag{3.23}$$

Cette séquence est modulée, transmise sur le canal et démodulée, et en entrée du décodeur, on reçoit la séquence binaire :

$$\mathbf{Y} = (Y_1, Y_2, ..., Y_n) \tag{3.24}$$

Nous supposerons également qu'en plus de la séquence $\mathbf{Y}$, nous recevons une séquence de n nombre positifs notée :

$$\alpha = (\alpha_1, \alpha_2, ..., \alpha_n) \tag{3.25}$$

Ces nombres positifs, appelés la mesure du canal, fourniront une mesure de la fiabilité des bits reçus. Ainsi, si $\alpha_i > \alpha_j$, le décodeur supposera que Y_i a plus de chance d'être correct que Y_j. Le décodeur qui utilise cette information n'est donc plus un décodeur à entrée dure.

Le décodeur incomplet (entrée dure) détermine le mot de code $\mathbf{X}^m = (X_1^m, X_2^m, ..., X_n^m)$ différent en un minimum d'emplacements de la séquence reçue $\mathbf{Y} = (Y_1, Y_2, ..., Y_n)$ à la condition que cette différence soit inférieure à $\left\lfloor \frac{d_{min}-1}{2} \right\rfloor$. La séquence contenant des 1 aux emplacements où $\mathbf{Y}$ et $\mathbf{X}^m$ diffèrent, la séquence d'erreurs, est donnée par :

$$\mathbf{Z}^m = \mathbf{Y} \oplus \mathbf{X}^m = (Y_1 \oplus X_1^m, Y_2 \oplus X_2^m, ..., Y_n \oplus X_n^m) \tag{3.26}$$

où la notation $\oplus$ désigne l'addition modulo-2. Si on définit le poids binaire de la séquence $\mathbf{Z}^m$ par :

$$W(\mathbf{Z}^m) = \sum_{i=1}^{n} Z_i^m \tag{3.27}$$

la fonction réalisée par le décodeur binaire incomplet est la recherche du mot de code (ou mot d'erreurs) qui satisfait :

$$W(\mathbf{Z}^m) \leq \left\lfloor \frac{d_{min} - 1}{2} \right\rfloor \tag{3.28}$$

Le décodeur incomplet trouvera un unique mot de code si l'inégalité est satisfaite ; sinon, il ne trouvera pas de mot de code. La décision sera correcte si le canal a effectivement introduit moins de $\left\lfloor \frac{d_{min}-1}{2} \right\rfloor$ erreurs.

Un décodeur complet peut être défini comme étant capable de trouver le mot de code $\mathbf{X}^m$qui satisfait :

$$\min_m W\left(\mathbf{Y} \oplus \mathbf{X}^m\right)$$

où m parcourt tous les mots de codes possibles. La différence avec le décodeur incomplet défini précédemment est qu'on peut obtenir un mot de code même si la séquence d'erreurs a un poids supérieur à $\left\lfloor \frac{d_{min}-1}{2} \right\rfloor$.

De la même manière, on peut définir un décodeur souple complet avec mesure de canal comme un décodeur capable de trouver le mot de code qui satisfait :

$$\min_m W_\alpha\left(\mathbf{Y} \oplus \mathbf{X}^m\right) \tag{3.29}$$

où dans ce cas il faut trouver la séquence d'erreur $\mathbf{Z}^m = \mathbf{Y} \oplus \mathbf{X}^m$de poids analogique minimum, le poids analogique d'un mot $\mathbf{Z}^m$ étant défini comme suit :

$$W_\alpha\left(\mathbf{Z}^m\right) = \sum_{i=1}^{n} \alpha_i Z_i^m \tag{3.30}$$

Le décodeur que nous allons décrire aura pour but de présenter des performances proches de celles de ce décodeur souple complet.

3.3.2.2 Algorithme de Chase

Le principe peut être illustré par le schéma suivant :

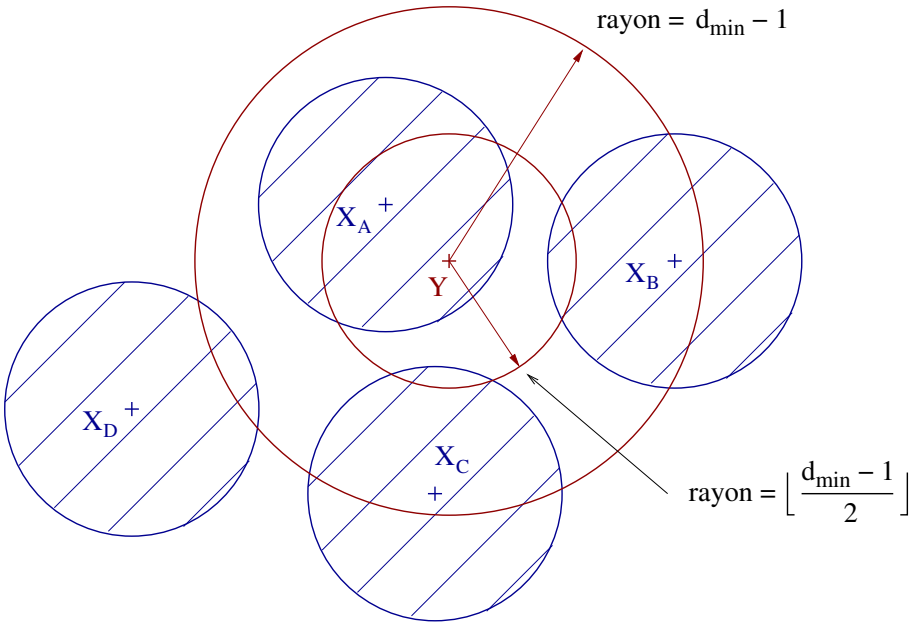


Figure 3.5: Principe du Chasing.

Ce schéma montre quatre mots de code $\mathbf{X}_A$, $\mathbf{X}_B$, $\mathbf{X}_C$, $\mathbf{X}_D$ et un mot reçu $\mathbf{Y}$, chacun étant entouré d'une sphère de rayon $\left\lfloor \frac{d_{min}-1}{2} \right\rfloor$. Dans ce cas précis, un décodeur hard donnerait pour unique mot d'erreurs $\mathbf{Z} = \mathbf{Y} \oplus \mathbf{X}_A$, $\mathbf{X}_A$étant le seul mot de code dans la sphère de rayon $\left\lfloor \frac{d_{min}-1}{2} \right\rfloor$ autour de $\mathbf{Y}$. Le but de la mesure de canal est d'utiliser le décodeur hard pour obtenir un ensemble relativement restreint de mots d'erreurs au lieu d'un seul, et ainsi de sélectionner celui de poids analogique minimum, comme défini précédemment.

L'ensemble des mots d'erreurs considéré est obtenu en perturbant la séquence reçue $\mathbf{Y}$ à l'aide de mots tests $\mathbf{T}$, séquences binaires contenant des 1 aux endroits où les bits de $\mathbf{Y}$ doivent être inversés. En additionnant ce mot test, modulo-2, au mot reçu, un nouveau mot :

$$\mathbf{Y}_T = \mathbf{Y} \oplus \mathbf{T} \tag{3.31}$$

est obtenu, et par décodage hard un nouveau mot d'erreurs $\mathbf{Z}'$ est obtenu. Le mot d'erreurs réel relatif à $\mathbf{Y}$ est donné par :

$$\mathbf{Z}_T = \mathbf{T} \oplus \mathbf{Z}' \tag{3.32}$$

$\mathbf{Z}_T$ peut être différent du mot d'erreurs du décodage classique, suivant que $\mathbf{Y}_T$ tombe ou non dans la sphère d'un nouveau mot de code. Pour cette classe de décodeurs, le mot reçu sera toujours perturbé dans une sphère de rayon $d_{min} - 1$, qui entoure $\mathbf{Y}$.

La figure 3.6 présente le fonctionnement du décodeur souple de Chase. Tous les algorithmes que nous présentons dans le paragraphe suivant utilisent strictement le même principe. Cependant, l'ensemble des mots tests utilisés dans chacun de ces algorithmes sera de plus en plus petit. Dans le cas où le décodeur dur utilisé ne décode que s'il trouve un mot à distance inférieure à $\left\lfloor \frac{d_{min}-1}{2} \right\rfloor$, il est possible qu'on ne trouve pas de mot d'erreurs pour un algorithme donné. Dans ce cas, le décodeur de Chase se contentera de ressortir le mot reçu, malgré le fait qu'il contienne des erreurs.

3.3.2.3 Différents algorithmes

Les différents algorithmes exposés ici diffèrent uniquement en la création des mots tests, et permettent des performances plus ou moins proches des performances optimales, et des quantités de calcul différentes en conséquence.

Algorithme 1

Pour cet algorithme un ensemble très important de mots tests est considéré. On considère l'ensemble des mots d'erreurs $\mathbf{Z}'$ dans la sphère de rayon $d_{min} - 1$ autour de la séquence reçue $\mathbf{Y}$. Ainsi, tous les mots d'erreurs de poids inférieur ou égal à $d_{min} - 1$ seront considérés. Etant donné que le mot d'erreurs sélectionné est déterminé par son poids analogique, il sera possible de sélectionner un mot d'erreurs qui contiendra plus de $\left\lfloor \frac{d_{min}-1}{2} \right\rfloor$ " 1 " et ainsi étendre la capacité de correction du code.

Un ensemble de mots tests suffisant mais non nécessaire, pour générer tous les mots d'erreurs de poids inférieur à d_{min} est donné par l'ensemble des mots $\mathbf{T}$ qui contiennent $\left\lfloor \frac{d_{min}}{2} \right\rfloor$ " 1 ". Le décodeur binaire étant capable de donner un mot d'erreurs de poids

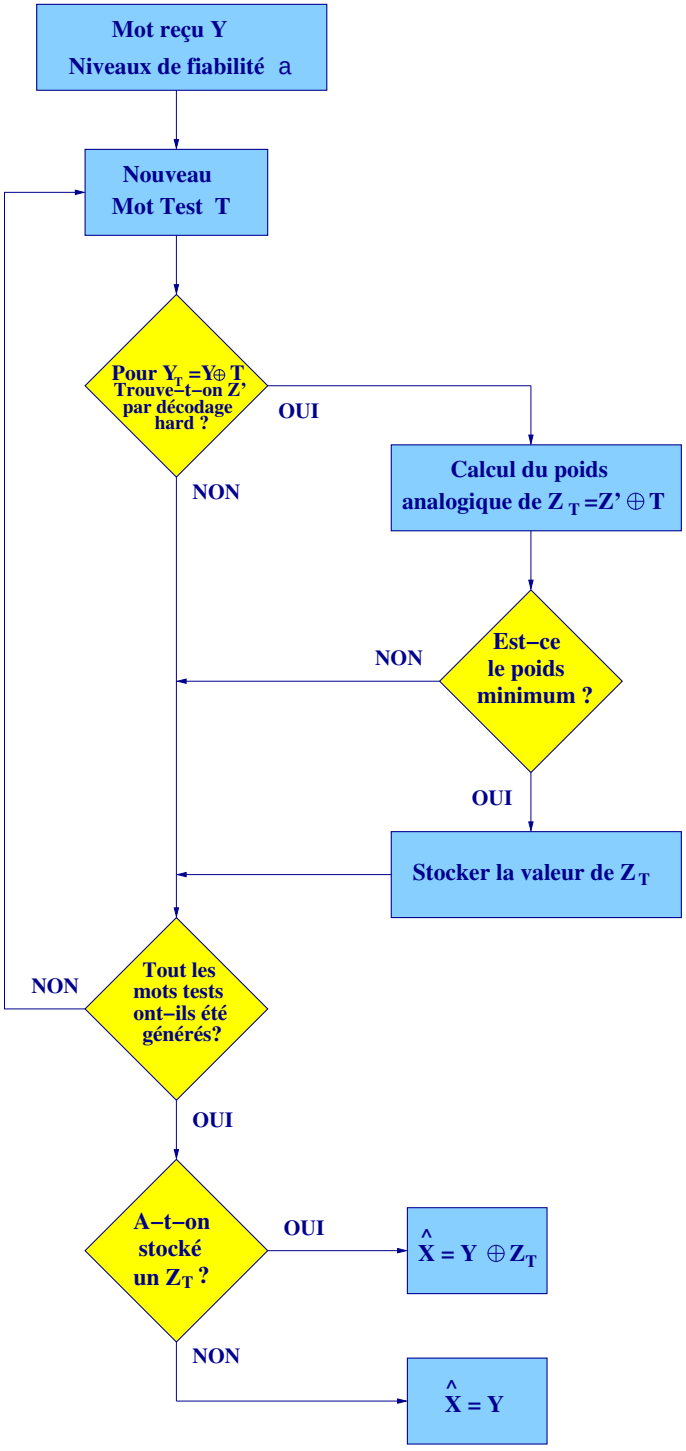


Figure 3.6: Algorithme de Chase.

maximum $\left\lfloor \frac{d_{min}-1}{2} \right\rfloor$, il sera possible, lorsqu'il est associé à un mot test approprié, de produire un mot d'erreurs de poids maximum $(d_{min} - 1)$.

Puisqu'il y a donc

$$C_n^{\lfloor d_{min}/2 \rfloor} = \frac{n!}{(n - \lfloor d_{min}/2 \rfloor)! \, (\lfloor d_{min}/2 \rfloor)!} \tag{3.33}$$

différents mots tests, cette méthode n'est applicable que pour les codes dont la distance minimale est faible. En fait, une importante diminution du nombre de mots tests et de la complexité peut être obtenue en éliminant les mots tests qui amènent à un même mot d'erreurs.

Algorithme 2

Pour cet algorithme, un ensemble beaucoup plus petit de mots tests est utilisé. Les mots d'erreurs $\mathbf{Z}'$ obtenus présentent au plus $\left\lfloor \frac{d_{min}-1}{2} \right\rfloor$ erreurs en dehors de l'ensemble qui contient les $\left\lfloor \frac{d_{min}}{2} \right\rfloor$ mesures de canal les plus faibles. Les mots d'erreurs ne contiennent alors pas plus de $(d_{min} - 1)$ erreurs, mais on ne teste plus tous les mots d'erreurs possibles de poids inférieur à $(d_{min} - 1)$.

Un ensemble de mots tests, qui génère tous les mots d'erreurs requis, est obtenu en laissant $\mathbf{T}$ prendre toutes les combinaisons de " 1 " situées dans les $\left\lfloor \frac{d_{min}}{2} \right\rfloor$ positions les moins fiables, c'est à dire les $\left\lfloor \frac{d_{min}}{2} \right\rfloor$ positions de mesure de canal les plus faibles. Etant donné qu'il y a $2^{\lfloor d_{min}/2 \rfloor}$ mots tests possibles, dont le mot tout à 0, il y a au maximum $2^{\lfloor d_{min}/2 \rfloor}$ mots d'erreurs considérés pour le décodage. Cet algorithme présente asymptotiquement les mêmes performances que l'algorithme 1 pour une complexité plus faible.

Algorithme 3

Cet algorithme est quasi identique à l'algorithme 2 mais on n'utilise plus que $\lfloor d_{min}/2 + 1 \rfloor$ mots tests au lieu de $2^{\lfloor d_{min}/2 \rfloor}$. Chaque mot test possède i " 1 " situés dans les i positions les moins fiables. Pour un code de distance minimale d paire, i prend les valeurs $i = 1, 3, \cdots, d_{min} - 1$ et $i = 0$. Quand d_{min} est impaire, $i = 0, 2, 4, \cdots, d_{min} - 1$.

Cet algorithme particulier présente le plus petit nombre possible de mots tests et donne les mêmes performances asymptotiques que les algorithmes 1 et 2. Il est néanmoins moins performant mais demande une puissance de calcul largement inférieure, ce qui le rend intéressant pour les applications où le code utilisé a une grande distance minimale.

Il existe d'autre types de décodages "par liste" [For66][TLFL01], basés sur les même principes, mais choisissant les mots de la liste suivant différents critères conduisant à différents compromis compléxité/performance. Néanmoins l'amélioration apportée par ces algorithmes, de manière analogue au décodage souple de l'algorithme de Viterbi pour les codes convolutifs [Vit67], bute dans le meilleur des cas à 3 dB environ de la limite de Shannon. Cet écart a été finalement comblé au bout d'un demi siècle de recherches intensives grâce à l'avènement des turbo codes [BGT93].

3.4 Turbo Codes Blocs : Algorithme de Pyndiah

Les turbo codes convolutifs ont été présentés pour la première fois en 1993 par C. Berrou [BG96][BGT93], montrant qu'il était possible d'approcher la limite prédite par C. Shannon. Depuis, de nombreux auteurs se sont intéressés aux turbo codes convolutifs et relativement peu aux turbo codes blocs, car les premières études dans ce domaine donnaient des performances médiocres dues à l'utilisation de décodeurs à entrée et sortie dures. Cependant, des travaux basés sur de petits codes blocs, utilisant les treillis de ces codes, présentés par J. Lodge [LYHH93] et J. Hagenauer [HOP96] donnèrent de bien meilleurs résultats.

Nous allons décrire ici un algorithme de décodage itératif imaginé par R. Pyndiah [PGPJ94][Pyn98], applicable à n'importe quel code produit construit à partir de codes blocs linéaires. Cet algorithme présente une complexité raisonnable, ce qui a permis la réalisation de circuits décodeurs. Il est basé sur des décodeurs Soft-Input/Soft-Output pour décoder les composants du code de manière à s'approcher des performances optimales à chaque itération. La sortie souple est une estimation de la log-vraisemblance calculée à partir des décisions binaires données par un décodeur de Chase.

3.4.1 Construction d'un Code Produit

Le concept des codes produits est très simple et relativement efficace pour construire de longs codes blocs à partir de deux codes blocs courts. Considérons deux codes blocs linéaires sous forme systématiques, $\mathbf{C}^1$ de paramètres (n_1, k_1, d_1) et $\mathbf{C}^2$ de paramètres (n_2, k_2, d_2), où n_i, k_i, d_i représentent respectivement la longueur, le nombre de bits informatifs et la distance minimale du code.

Le code produit $\mathbf{P} = \mathbf{C}^1 \otimes \mathbf{C}^2$ est obtenu de la manière suivante :

1. Placer $(k_1 \times k_2)$ bits d'information dans un tableau à k_1 lignes et k_2 colonnes ;

2. Coder les k_1 lignes par $\mathbf{C}^2$;

3. Coder les n_2 colonnes par $\mathbf{C}^1$.

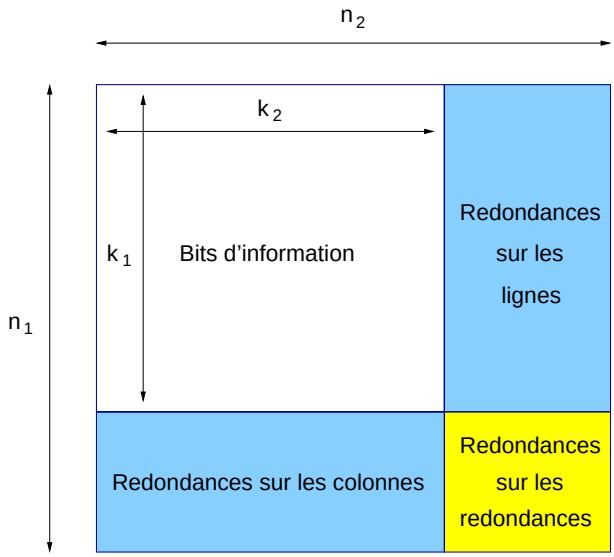


Figure 3.7: Construction d'un code produit.

La figure 3.7 donne un résumé de la construction du code.

Les paramètres du code produit $\mathbf{P}$ sont $n = n_1 \times n_2$, $k = k_1 \times k_2$, $d = d_1 \times d_2$, et le rendement de codage R est $R = R_1 \times R_2$, où R_i est le rendement de codage du code $\mathbf{C}^i$. Ainsi, on peut construire de longs codes blocs avec une grande distance minimale en combinant deux codes plus petits. On peut remarquer que, par linéarité, toutes les lignes de $\mathbf{P}$ sont des mots de code de $\mathbf{C}^2$ et toutes les colonnes sont des mots de code de $\mathbf{C}^1$.

Ces codes peuvent être décodés en décodant les lignes puis les colonnes. Cependant, pour atteindre les performances optimales, il faut utiliser le décodage souple avec une sortie souple. On pourra ainsi effectuer plusieurs itérations pour approcher les performances optimales du code produit.

3.4.2 Décodage à entrée souple d'un code bloc linéaire

Considérons la transmission d'éléments binaires codés par un code bloc $\mathbf{C}$ de paramètres (n, k, d) sur un canal Gaussien utilisant en bande de base des symboles binaires $\{-1, +1\}$. L'observation $\mathbf{R} = (r_1, \cdots, r_l, \cdots, r_n)$ à la sortie du canal gaussien pour un mot de code émis $\mathbf{E} = (e_1, \cdots, e_l, \cdots, e_n)$ est :

$$\mathbf{R} = \mathbf{E} + \mathbf{G} \tag{3.34}$$

où chacune des composantes g_l de $\mathbf{G} = (g_1, \cdots, g_l, \cdots, g_n)$ est un bruit blanc gaussien additif de variance σ^2. La décision optimum sur le mot de code transmis est donnée par (3.35).

$$\mathbf{D} = \mathbf{C}^i \text{ si } \left|\mathbf{R} - \mathbf{C}^i\right|^2 \leq \left|\mathbf{R} - \mathbf{C}^l\right|^2 \quad \forall l \in \left[1, 2^k\right] \tag{3.35}$$

,

où $\mathbf{C}^i = (c_1^i, \cdots, c_l^i, \cdots, c_n^i)$ est le $i^{ème}$ mot de code de $\mathbf{C}$ et

$$\left|\mathbf{R} - \mathbf{C}^i\right|^2 = \sum_{l=1}^{n} \left(r_l - c_l^i\right)^2 \tag{3.36}$$

est le carré de la distance euclidienne entre $\mathbf{R}$ et $\mathbf{C}^i$. Pour réaliser cette opération, on utilise un décodeur de Chase, défini au chapitre précédent. Pour ce qui est du code produit, cet algorithme permet d'obtenir une décision $\mathbf{D}$ sur un mot élémentaire $\mathbf{R}$ composant le code produit. Dans le but d'itérer le processus de décodage, il faut calculer la fiabilité des décisions données par l'algorithme de Chase avant de décoder les lignes (respectivement les colonnes).

3.4.3 Fiabilité de la décision D donnée par le Chasing

Une fois que l'on a déterminé la décision $\mathbf{D}$ pour une ligne (respectivement une colonne) du code produit, il faut calculer la fiabilité de chaque bit du vecteur $\mathbf{D}$ afin de fournir une information souple en sortie du décodeur. La fiabilité d'une décision d_j est définie en utilisant le LLR (log-vraisemblance) du symbole transmis e_j, qui est donné par :

$$\Lambda(d_j) = \ln\left(\frac{\Pr\left\{e_j = +1/\mathbf{R}\right\}}{\Pr\left\{e_j = -1/\mathbf{R}\right\}}\right) \tag{3.37}$$

Il faut alors tenir compte du fait que $\mathbf{D}$ est un des 2^k mots de code de $\mathbf{C}$. Ainsi, en considérant les différents mots de code de $\mathbf{C}$, on peut écrire :

$$\Pr\left\{e_j = +1/\mathbf{R}\right\} = \sum_{\mathbf{C}^i \in \mathbf{S}_j^{+1}} \Pr\left\{\mathbf{E} = \mathbf{C}^i/\mathbf{R}\right\} \tag{3.38}$$

où $\mathbf{S}_j^{+1}$ est l'ensemble des mots de code $\mathbf{C}^i$ tel que $c_j^i = +1$ et on peut de la même manière écrire l'équation (3.39) :

$$\Pr\left\{e_j = -1/\mathbf{R}\right\} = \sum_{\mathbf{C}^i \in \mathbf{S}_j^{-1}} \Pr\left\{\mathbf{E} = \mathbf{C}^i/\mathbf{R}\right\} \tag{3.39}$$

où $\mathbf{S}_j^{-1}$ est l'ensemble des mots de code $\mathbf{C}^i$ tel que $c_j^i = -1$.

En appliquant les règles de Bayes aux équations précédentes, et en supposant que les différents mots de code sont uniformément distribués, nous obtenons pour $\Lambda(d_j)$ l'expression suivante :

$$\Lambda(d_j) = \ln\left(\frac{\sum_{\mathbf{C}^i \in \mathbf{S}_j^{+1}} \Pr\left\{\mathbf{R}/\mathbf{E} = \mathbf{C}^i\right\}}{\sum_{\mathbf{C}^i \in \mathbf{S}_j^{-1}} \Pr\left\{\mathbf{R}/\mathbf{E} = \mathbf{C}^i\right\}}\right) \tag{3.40}$$

où

$$\Pr\left\{\mathbf{R}/\mathbf{E} = \mathbf{C}^i\right\} = \left(\frac{1}{\sqrt{2\pi\sigma}}\right)^n \exp\left(-\frac{\left|\mathbf{R} - \mathbf{C}^i\right|^2}{2\sigma^2}\right) \tag{3.41}$$

est la densité de probabilité de $\mathbf{R}$ conditionnée à $\mathbf{E}$. Cette fonction décroît exponentiellement avec la distance euclidienne entre $\mathbf{R}$ et $\mathbf{C}^i$. Soient $\mathbf{C}^{+1(j)}$ et $\mathbf{C}^{-1(j)}$ les mots de code, respectivement, dans $\mathbf{S}_j^{+1}$ et $\mathbf{S}_j^{-1}$, à distance euclidienne minimum de $\mathbf{R}$. En combinant les deux équations précédentes, on obtient :

$$\Lambda\left(d_j\right) = \frac{1}{2\sigma^2}\left(\left|\mathbf{R} - \mathbf{C}^{-1(j)}\right|^2 - \left|\mathbf{R} - \mathbf{C}^{+1(j)}\right|^2\right) + \ln\left(\frac{\sum_i \mathbf{A}_i}{\sum_i \mathbf{B}_i}\right) \tag{3.42}$$

où

$\mathbf{A}_i = \exp\left(\frac{\left|\mathbf{R}-\mathbf{C}^{+1(j)}\right|^2 - \left|\mathbf{R}-\mathbf{C}^{i}\right|^2}{2\sigma^2}\right) \leq 1$ avec $\mathbf{C}^i \in \mathbf{S}_j^{+1}$

et

$\mathbf{B}_i = \exp\left(\frac{\left|\mathbf{R}-\mathbf{C}^{-1(j)}\right|^2 - \left|\mathbf{R}-\mathbf{C}^{i}\right|^2}{2\sigma^2}\right) \leq 1$ avec $\mathbf{C}^i \in \mathbf{S}_j^{-1}$

Pour les forts Rapport Signal sur Bruit, $\sum_i A_i$ et $\sum_i B_i$ tendent vers 1 et ainsi le second terme tend vers 0. En négligeant ce terme, on obtient une approximation pour le LLR de la décision d_j :

$$\Lambda'\left(d_j\right) = \frac{1}{2\sigma^2}\left(\left|\mathbf{R} - \mathbf{C}^{-1(j)}\right|^2 - \left|\mathbf{R} - \mathbf{C}^{+1(j)}\right|^2\right) \tag{3.43}$$

En développant cette expression, on obtient :

$$\Lambda'\left(d_j\right) = \frac{2}{\sigma^2}\left(r_j + \sum_{l=1, l\neq j}^{n} r_l c_l^{+1(j)} p_l\right) \tag{3.44}$$

où

$$p_l = \begin{cases} 0, & si\ c_l^{+1(j)} = c_l^{-1(j)} \\ 1, & si\ c_l^{+1(j)} \neq c_l^{-1(j)} \end{cases} \tag{3.45}$$

On peut normaliser $\Lambda'\left(d_j\right)$ par rapport à $2/\sigma^2$, ce qui permet d'exprimer simplement la log-vraisemblance en sortie comme la somme de l'entrée et d'un terme w_j :

$$r'_j = r_j + w_j \tag{3.46}$$

avec

$$w_j = \sum_{l=1, l\neq j}^{n} r_l c_l^{+1(j)} p_l \tag{3.47}$$

Le LLR normalisé est utilisé comme sortie souple du décodeur. Il a le même signe que d_j et sa valeur absolue représente la fiabilité de la décision. On observe que r'_j est obtenu par r_j additionné à un terme w_j fonction des deux mots de code à distance minimum de $\mathbf{R}$. Le terme w_j est un terme de correction appliqué aux données d'entrée appelé information extrinsèque. Il représente l'information apportée par le codage. Nous allons maintenant décrire l'algorithme utilisé pour calculer la fiabilité de la décision.

3.4.4 Calcul de la sortie souple du décodeur

Le calcul de la fiabilité de la décision d_j à la sortie du décodeur à entrée souple nécessite deux mots de code $\mathbf{C}^{+1(j)}$ et $\mathbf{C}^{-1(j)}$; évidemment, $\mathbf{D}$ est un de ces deux mots de code et il nous faut trouver le deuxième, que nous appellerons $\mathbf{C}^c$. $\mathbf{C}^c$ peut être vu comme un mot de code concurrent pour $\mathbf{D}$ à distance euclidienne minimale de $\mathbf{R}$ avec $c_j^c \neq d_j$.

Pour trouver le mot de code $\mathbf{C}^c$, il faut augmenter la taille de l'espace parcouru par l'algorithme de Chase. Dans ce but, il faut augmenter le nombre p de bits les moins fiables et donc le nombre de mots test. Il est évident que la probabilité de trouver $\mathbf{C}^c$ augmente avec p. Cependant, la complexité du décodeur augmente exponentiellement avec p et il faut donc trouver un compromis. Cela implique que dans certains cas on ne trouvera pas $\mathbf{C}^c$ et qu'il faudra utiliser une autre méthode. La solution initialement proposée par Pyndiah [Pyn98] est la suivante :

$$r'_j = \beta \times d_j \tag{3.48}$$

où β est une constante positive à optimiser. Cette solution est simple mais cohérente avec le fait que, si on ne trouve pas $\mathbf{C}^c$, c'est qu'il est à une distance relativement grande de $\mathbf{R}$, et en tout cas supérieure à celle parcourue par le Chasing, et que par conséquent la décision d_j a une grande probabilité d'être bonne.

3.4.5 Turbo décodage des codes produits

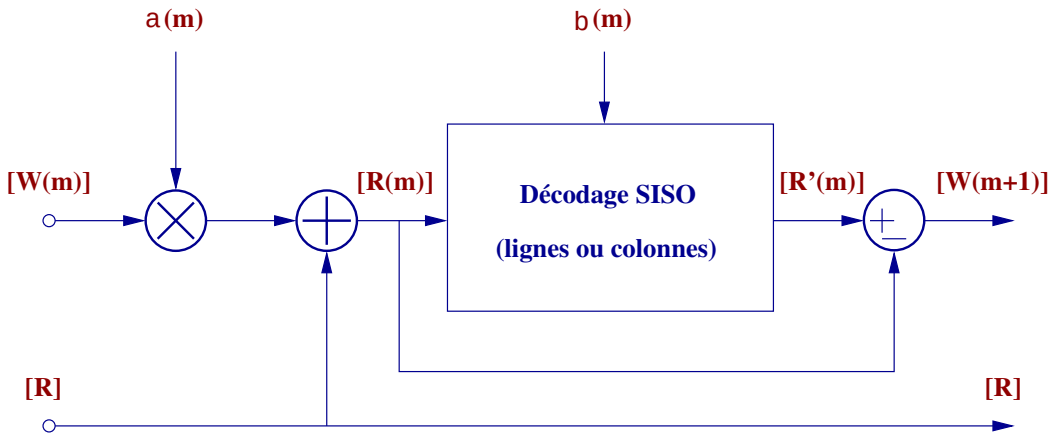


Figure 3.8: Turbo décodeur élémentaire (1/2 itération)

Considérons le décodage des lignes (respectivement des colonnes) du code produit $\mathbf{P}$ transmis sur un canal gaussien. A la réception de la matrice [$\mathbf{R}$] correspondant à un mot de code produit transmis [$\mathbf{E}$], le premier décodeur calcule le décodage souple des lignes (respectivement des colonnes) en utilisant la matrice d'entrée [$\mathbf{R}$]. Par soustraction entre l'entrée souple [$\mathbf{R}$] et la sortie souple [$\mathbf{R}'$], on obtient l'information extrinsèque [$\mathbf{W}$ ($\mathbf{2}$)], le paramètre 2 indiquant que l'on considère l'information extrinsèque pour le deuxième

décodage de [**R**], calculée pendant le premier décodage. L'entrée souple du décodage des colonnes (respectivement des lignes) au second décodage de [**R**] est donnée par :

$$[\mathbf{R}(2)] = [\mathbf{R}] + \alpha(2)[\mathbf{W}(2)] \tag{3.49}$$

où $\alpha(2)$ est un facteur de pondération qui tient compte du fait que l'écart type des éléments de [**R**] et de [**W**] sont différents. La variance de l'information extrinsèque est très grande pendant les premières itérations et diminue ensuite. Ce facteur permet aussi de réduire l'effet de l'information extrinsèque pendant les premières étapes du décodage où le taux d'erreur binaire (TEB) est relativement important. Il prend une faible valeur dans les premières itérations puis augmente progressivement. Le processus décrit ci-dessus est donc obtenu en cascadant des décodeurs élémentaires tel que sur la figure 3.8.

Les constantes α et β varient à chaque itération, en commençant par une valeur faible pour atteindre 1 au bout de quelques itérations. R. Pyndiah a initialement déterminé les coefficients suivants :

$\alpha(m) = [0;0,2;0,3;0,5;0,7;0,9;1,0;1,0]$

$\beta(m) = [0,2;0,4;0,6;0,8;1,0;1,0;1,0;1,0]$

Pour un nombre d'itérations plus important, les coefficients manquants seront tous à 1.

3.4.6 Amélioration concernant l'utilisation de β

Dans [AP00], une technique visant à éliminer le choix des paramètres β dans l'algorithme est proposée. Ces paramètres sont en effet génants car ils sont fixés de manière arbitraire et se trouvent donc être indépendants des fiabilités des bits reçus. Or ils sont souvent utilisés au cours des itérations du turbo décodage. La conséquence est une dégradation du taux d'erreur binaire car l'algorithme est de ce fait sous optimal. La solution proposée dans [AP00] consiste à tirer parti de l'information que l'on possède pour obtenir une meilleure estimation de cette fiabilité. On définit alors l'estimation de $\Lambda'(d_j)$ comme étant :

$$\left|r'_j\right| = |r_j| + \Sigma_{k=x}^{y} |r_k| \tag{3.50}$$

où x et y sont choisis parmi les bits les moins fiables du vecteur ligne ou colonne considéré. Cette solution donne un gain de l'ordre de 0, 1 à 0, 2 dB par rapport à la solution où les β sont fixés et permet donc de se rapprocher de la courbe théorique. Le principal avantage est qu'il n'est plus nécessaire de régler ce paramètre et que cette solution fonctionne très correctement quelque soit le code et la constellation.

3.4.7 Mise en oeuvre des turbo codes blocs

Ce paragraphe a pour but de proposer une architecture intéressante [GBBV04a] pour le décodeur Soft-In/Soft-Out (SISO) de Pyndiah. Une grande économie de mémoire est ici réalisée car il est inutile de conserver les vecteurs testés par l'algorithme de Chase sous contrainte d'ordonner les poids de manière appropriée. Dans un premier temps je rappelerai l'implémentation classique de l'algorithme de Pyndiah, ainsi que le principe

du Fast Chasing [HHM01]. Puis j'exposerai le principe de l'algorithme simplifié dans son détail en justifiant les opérations effectuées.

3.4.7.1 Rappels sur le calcul du "poids" et de la sortie

Soit $\mathbf{C}^{+1(j)} = \left\{c_0^{+1(j)}, \cdots, c_n^{+1(j)}\right\}$ et $\mathbf{C}^{-1(j)} = \left\{c_0^{-1(j)}, \cdots, c_n^{-1(j)}\right\}$ les mots (mappés sur $\{+1, -1\}$) à distance minimale de $\mathbf{R}$ avec pour contrainte que le j^{eme} bit soit respectivement égal à $+1$ et -1.
On rappelle la définition des poids analogiques (distance euclidienne par rapport au mot reçu) :

$$m^{+1(j)} = \left\|\mathbf{R} - \mathbf{C}^{+1(j)}\right\|^2 \text{ et } m^{-1(j)} = \left\|\mathbf{R} - \mathbf{C}^{-1(j)}\right\|^2 \tag{3.51}$$

et on sait que la log-vraisemblance normalisée est donnée par :

$$r'_j = \frac{m^{-1(j)} - m^{+1(j)}}{4} \tag{3.52}$$

or :

$$m^{+1(j)} = \sum_{i=0}^{n} \left(r_i - c_i^{+1(j)}\right)^2 = \sum_{i=0}^{n} \left(r_i^2 + 1 - 2r_i c_i^{+1(j)}\right) \tag{3.53}$$

et de même :

$$m^{-1(j)} = \sum_{i=0}^{n} \left(r_i^2 + 1 - 2r_i c_i^{-1(j)}\right) \tag{3.54}$$

donc :

$$r'_j = \left(-\frac{1}{2}\sum_{i=0}^{n} r_i c_i^{-1(j)}\right) - \left(-\frac{1}{2}\sum_{i=0}^{n} r_i c_i^{+1(j)}\right) \tag{3.55}$$

Si on définit un nouveau poids analogique P pour le mot $\mathbf{C} = \{c_0, \cdots, c_n\}$ par :

$$P(\mathbf{C}) = -\frac{1}{2}\sum_{i=0}^{n} r_i c_i \tag{3.56}$$

alors on conserve le même ordre de classement qu'avec les définitions précédentes par distance euclidienne (3.51). Cette quantité peut donc être utilisée pour classer les vecteurs testés par le Chasing.

3.4.7.2 Implémentation de Pyndiah

L'implémentation classique de cette algorithme, telle qu'elle est présentée par Pyndiah [Pyn98], peut être résumée de la manière suivante pour une ligne ou une colonne :

- Algorithme de Chase avec décodage par l'utilisation d'un algorithme de type Berlekamp-Massey ou Peterson-Gorenstein-Zierler et stockage en mémoire des mots obtenus et de leurs poids ;
- Recherche de la décision dure $\mathbf{C}^d = \mathbf{D}$ (distance euclidienne minimale) parmi ces mots ;

- Pour chaque bit j, recherche du mot concurrent $\mathbf{C}^c$ à distance euclidienne minimale de $\mathbf{R}$ tel que $c_j^d \neq c_j^c$ et calcul de la fiabilité $f_j = \left[\frac{m^c - m^d}{4}\right]$ où m^c est le poids de $\mathbf{C}^c$ et m^d le poids de $\mathbf{C}^d$;
- Calcul des informations extrinsèques par : $w_j = \left[f_j - c_j^d \cdot r'_j\right] \cdot c_j^d$

Cette implantation nécessite donc la mise en mémoire de tous les mots concurrents de n bits pour chaque ligne ou colonne décodée. De plus, après la boucle du Chasing, deux autres boucles sont nécessaires pour arriver au résultat final : une boucle de recherche de la décision de Chase et une boucle de recherche du mot concurrent pour chacun des bits.

Nous allons voir dans la partie 3.4.7.4 qu'il est possible de s'affranchir du stockage des mots obtenus lors du Chasing et de faire l'ensemble des opérations en une seule boucle.

3.4.7.3 Algorithme de Chase Rapide : Fast-Chasing

On considère un code de Hamming étendu par un bit de parité y_0. Cet algorithme, présenté dans [HHM01], permet de simplifier l'opération de Chasing pour obtenir tous les mots concurrents en parcourant les vecteurs de test par un comptage de type Gray. Cela permet de simplifier l'expression du syndrome pour chaque itération en tirant partie des propriétés des codes blocs linéaires dans le cas d'un code 1-correcteur. Le calcul du poids est également simplifié car la mise à jour est simple quand on considère qu'un seul bit a changé (équation (3.58)).

Soient :

- $\mathbf{H}^T$ la matrice de controle du code BCH 1-correcteur considéré. Soit m l'ordre du corps de Galois, $\mathbf{H}^T$ possède m colonnes et $2^m - 1 = n$ lignes. Le syndrome est $\mathbf{S} = \mathbf{Y}\mathbf{H}^T$. $\mathbf{H}_i^T$ designe la i^{eme} ligne de $\mathbf{H}^T$. Le bit de parité ne sera pas pris en compte dans le calcul du syndrome, mais controlé aprés coup. On remarque que si l'on change un seul bit i de $\mathbf{Y}$, le nouveau syndrome est obtenu simplement additionnant à l'ancien syndrome (modulo-2) le vecteur $\mathbf{H}_i^T$ correspondant à ce bit. Cette propriété est exploitée par la suite ; en effet, un comptage de Gray pour passer en revue les vecteurs tests permet de ne changer qu'un seul bit par itération et de ce fait simplifie énormément le calcul du syndrome.
- $\mathbf{R} = \{r_0, r_1, \cdots, r_n\}$ l'entrée souple et $\mathbf{R}' = \{r'_0, r'_1, \cdots, r'_n\}$ la sortie souple du décodeur SISO.
- $\mathbf{Y} = \{y_0, y_1, \cdots, y_n\}$ la décision dure du mot $\mathbf{R}$ à l'entrée du décodeur en prenant $y_i \in \{0, 1\}$. $\mathbf{Y}^{bm} = \left\{y_0^{bm}, \cdots, y_n^{bm}\right\}$ est le mot testé à chaque itérations du Chasing (il parcourt l'espace autour du mot reçu par un comptage Gray) et $\mathbf{Y}' = \{y'_0, \cdots, y'_n\}$ est le mot obtenu par décodage hard.
- $Poids^{bm}$ et $Poids$ les poids analogiques de $\mathbf{Y}^{bm}$ et $\mathbf{Y}'$ respectivement.

- **Bm** $= \{Bm_1, \cdots, Bm_{2^p-1}\}$ les numéros des bits modifiés d'un mot au suivant à partir du mot reçu pour parcourir l'espace autour de celui-ci dans le Chasing. L'opération est équivalente à compter en Binaire Gray sur les vecteurs tests. p est le nombre de positions les moins fiables prises en compte dans le Chasing. Il est inutile de compter le bit de parité dans les positions les moins fiables quelle que soit sa valeur (il n'intervient pas dans le decodage hard du BCH). Par exemple, pour $p = 3$, avec un code de longueur 7, si les positions les moins fiables sont $\{3, 5, 6\}$ alors **Bm** $= \{3, 5, 3, 6, 3, 5, 3\}$. Dans ce cas, pour un mot reçu $\mathbf{Y} = \{0, 0, 0, 0, 0, 0, 0\}$, $\mathbf{Y}^{bm}$ vaut successivement :

$$\begin{array}{ccccccc} \{0 & 0 & 0 & 0 & 0 & 0 & 0\} \\ \{0 & 0 & 1 & 0 & 0 & 0 & 0\} \\ \{0 & 0 & 1 & 0 & 1 & 0 & 0\} \\ \{0 & 0 & 0 & 0 & 1 & 0 & 0\} \\ \{0 & 0 & 0 & 0 & 1 & 1 & 0\} \\ \{0 & 0 & 1 & 0 & 1 & 1 & 0\} \\ \{0 & 0 & 1 & 0 & 0 & 1 & 0\} \\ \{0 & 0 & 0 & 0 & 0 & 1 & 0\} \end{array}$$

- $\oplus$ désigne l'addition des bits modulo-2 et pour des vecteurs elle s'effectue bit à bit.

De plus, si l'on ne change lors du Chasing que le k^{eme} symbole du mot $\mathbf{C}$ pour obtenir le mot $\mathbf{C}'$, on peut deduire le poids (3.56) de $\mathbf{C}'$ à partir du poids de $\mathbf{C}$ car :

$$P(\mathbf{C}') = -\frac{1}{2}\sum_{i=0}^{n} r_i c'_i = -\frac{1}{2}\sum_{i=0}^{n} r_i c_i + \frac{1}{2} r_k c_k - \frac{1}{2} r_k c'_k \tag{3.57}$$

et on sait que $c'_k = -c_k$, donc :

$$P(\mathbf{C}') = P(\mathbf{C}) - r_k c'_k \tag{3.58}$$

Or dans le Fast Chasing, on ne change qu'un seul bit à la fois à chaque étape. En effet, un seul bit est changé pour obtenir un nouveau vecteur de test à partir du précédent. De plus, le mot de test obtenu par décodage du vecteur de test est donné également par la modification d'un seul bit à la fois (correction puis parité). Ce bit de parité est d'ailleurs corrigé dans l'unique but de mettre à jour le poids, sans quoi il n'aurait pas d'utilité, ce qui apporte de l'information supplémentaire au décodage souple. On peut donc obtenir tous les poids à partir du poids du mot de départ en utilisant la relation (3.58). Le fait d'attribuer un poids nul au mot de départ ne change pas l'ordre du classement.

Voici donc le descriptif de l'algorithme :

Initialisation de l'algorithme

Variables

On initialise les variables utilisées par l'algorithme de la manière suivante :
$\mathbf{Y}^{bm} = \mathbf{Y}$ et $Poids^{bm} = 0$ (le premier mot testé par l'algorithme est le mot reçu, et le poids est considéré comme nul pour le mot reçu)
$\mathbf{Y}' = \mathbf{Y}$ et $Poids = 0$ ($\mathbf{Y}'$ est initialisé à la valeur du premier mot testé avant d'être décodé dans la prochaine étape)

Décodage hard

Calcul du sydrome : $\mathbf{S} = \mathbf{Y}\mathbf{H}^T$

Si $\mathbf{S} \neq 0$ alors $\Rightarrow$
Détermination de la position de l'erreur e par tableau de correspondance puis correction par :
$y'_e \oplus 1 \rightarrow y'_e$
et mise à jour du poids par :
$Poids - (2\,y'_e - 1)\,r_e \rightarrow Poids.$

Correction du bit de parité

Calcul de $b = y'_1 \oplus \cdots \oplus y'_n$

Si $b \neq y'_0$ alors $\Rightarrow$
correction par :
$y'_0 \oplus 1 \rightarrow y'_0$
et mise à jour du poids par :
$Poids - (2\,y'_0 - 1)\,r_0 \rightarrow Poids.$

On possède à ce moment le mot $\mathbf{Y}'$ obtenu par décodage hard du mot reçu $\mathbf{Y}$ et son poids analogique. C'est le premier mot de code obtenu par Chasing et on le stocke en mémoire.

Itérations de l'algorithme

A chaque nouvelle itération, les variables conservent les valeurs qu'elles avaient à la fin de la précédente itération et pour $i = 1$ à $i = 2^p - 1$ on effectue les opérations ci-aprés :

Modification du bit pour obtenir le mot suivant

On obtient le mot à tester $\mathbf{Y}^{bm}$ pour l'itération courante à partir de celui de l'itération précédente par la modification d'un seul bit determiné par le vecteur $\mathbf{Bm}$:

$y^{bm}_{\mathbf{Bm}(i)} \oplus 1 \rightarrow y^{bm}_{\mathbf{Bm}(i)}$

On met à jour le poids du vecteur testé :
$Poids^{bm} - (2\,y'_{\mathbf{Bm}(i)} - 1)r_{\mathbf{Bm}(i)} \rightarrow Poids^{bm}$

$\mathbf{Y'} = \mathbf{Y}^{bm}$ et $Poids = Poids^{bm}$ (Y' est initialisé à la valeur du vecteur testé avant d'être décodé)

Décodage hard

Le nouveau syndrome peut être déduit très simplement du précédent car un seul bit a été modifié : $\mathbf{S} \oplus \mathbf{H}^T_{\mathbf{Bm}(i)} \rightarrow \mathbf{S}$

Si $\mathbf{S} \neq 0$ alors $\Rightarrow$
Détermination de la position de l'erreur e par tableau de correspondance puis correction par :
$y'_e \oplus 1 \rightarrow y'_e$
et mise à jour du poids par :
$Poids - (2\,y'_e - 1)\,r_e \rightarrow Poids.$

Correction du bit de parité

Calcul de $b = y'_1 \oplus \cdots \oplus y'_n$

Si $b \neq y'_0$ alors $\Rightarrow$
correction par :
$y'_0 \oplus 1 \rightarrow y'_0$
et mise à jour du poids par :
$Poids - (2\,y'_0 - 1)\,r_0 \rightarrow Poids.$

On a alors le mot $\mathbf{Y'}$ produit par le décodage hard du mot testé $\mathbf{Y}^{bm}$ et son poids analogique. On sauvegarde alors le mot de code de Chase courant $\mathbf{Y'}$ ainsi que son poids. L'indice i est incrémenté et on passe à l'itération suivante (modification du bit pour obtenir le mot suivant).

Cet algorithme permet donc de parcourir simplement tous les mots concurrents et d'obtenir au passage les poids analogiques associés. On calcule ensuite les fiabilités de la même manière qu'au 3.4.7.2 en utilisant les mots de code précédemment stockés.

3.4.7.4 Architecture simplifiée avec économie de mémoire

Nous proposons ici une amélioration intéressante [GBBV04a] en terme de quantité de mémoire nécessaire à la mise en oeuvre du turbo décodeur.

Le résultat en sortie peut être obtenu par soustraction entre deux poids car, comme le montre l'équation (3.55) :

$$r'_j = P\left(\mathbf{C}^{-1(j)}\right) - P\left(\mathbf{C}^{+1(j)}\right) \tag{3.59}$$

On peut donc utiliser directement le poids défini par (3.56) pour le calcul de la sortie, et ce en conservant à chaque fois la distance minimale avec la contrainte que le j^{eme} bit

soit à $+1$ ou -1 pour $j = 0 \cdots n$. La valeur du poids du mot reçu $\mathbf{Y}$ à partir duquel tous les poids sont déduits sera soustrait à lui même et par conséquent, il n'influence pas la valeur de la sortie. On peut donc attribuer un poids nul à la décision dure $\mathbf{Y}$ sur le vecteur recu (ce qui ne change pas l'ordre du classement) et en déduire tous les poids. **Il n'est plus necessaire alors de conserver les mots de test décodés passés en revue, mais seulement leurs poids**. Il suffit de classer le poids de chaque mot de test décodé en fonction de la valeur de chaque bit du mot et en ne conservant que le plus faible dans chaque cas. Un vecteur contiendra les poids $P\left(\mathbf{C}^{+1(j)}\right)$ des $\mathbf{C}^{+1(j)}$ et un autre les poids $P\left(\mathbf{C}^{-1(j)}\right)$ des $\mathbf{C}^{-1(j)}$ pour $j = 0 \cdots n$. La sortie sera alors obtenue simplement, puisqu'en faisant la différence des poids, on retrouve l'expression de la sortie souple. L 'algorithme global est décrit par la figure 3.9. Les fonctions 2 et 3 sont celles de l'algorithme de Chase rapide. L'opération 4 de classement est détaillée sur la figure 3.10.

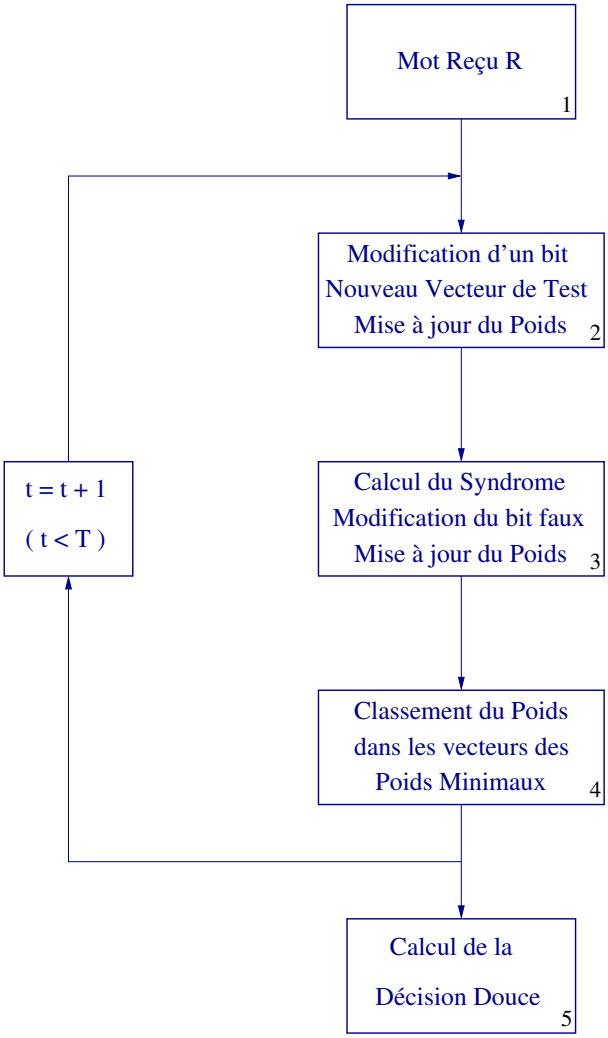


Figure 3.9: Algorithme Simplifié

L'algorithme devient alors le suivant :

Initialisation

On définit

$\mathbf{PM}^{(1)} = \left\{ PM_0^{(1)}, \cdots, PM_j^{(1)}, \cdots, PM_n^{(1)} \right\}$

et

$\mathbf{PM}^{(0)} = \left\{ PM_0^{(0)}, \cdots, PM_j^{(0)}, \cdots, PM_n^{(0)} \right\}$

la liste des poids des mots du code à distance minimum du mot recu avec le $j^{ème}$ bit respectivement à 1 et à 0.

Pour $j = 0 \cdots n$, on initialise les vecteurs contenant les poids minimaux :

$PM_j^{(1)} = +\infty$ et $PM_j^{(0)} = +\infty$

Classement et sauvegarde des poids minimaux A la fin de la première itération du Fast Chasing, on classe le poids du premier mot de test obtenu $\mathbf{Y'}$, dans les vecteurs de sauvegarde des poids minimaux en fonction des valeurs de ses bits. Il est forcément le poids minimum pour le moment, puisqu'il est le premier testé et on réalise donc l'opération suivante :

Pour $j = 0 \cdots n$,

Si $y'_j = 0 \Rightarrow PM_j^{(0)} = Poids$ (c.a.d que $\mathbf{Y'}$ est considéré pour le moment comme le mot ayant le bit j égal à 0 à distance minimum du mot reçu)

Si $y'_j = 1 \Rightarrow PM_j^{(1)} = Poids$ (c.a.d que $\mathbf{Y'}$ est considéré pour le moment comme le mot ayant le bit j égal à 1 à distance minimum du mot reçu)

Ensuite, pour **chacune des itérations**, on classe alors le poids actuel obtenu par l'itération de Fast Chasing courante dans les tableaux de sauvegarde des poids minimaux, dans $\mathbf{PM}^{(1)}$ ou dans $\mathbf{PM}^{(0)}$ en fonction de la valeur de chacun des bits, et seulement si le poids actuel est inférieur au poids précédemment stocké :

Pour $j = 0 \cdots n$,

Si $y'_j = 0$ et $Poids < PM_j^{(0)} \Rightarrow PM_j^{(0)} = Poids$

Si $y'_j = 1$ et $Poids < PM_j^{(1)} \Rightarrow PM_j^{(1)} = Poids$

Aprés avoir passé en revu tout les vecteurs de test, on aura :

$$\begin{aligned} PM_j^{(1)} &= P\left(\mathbf{C}^{+1(j)}\right) \\ PM_j^{(0)} &= P\left(\mathbf{C}^{-1(j)}\right) \end{aligned}$$

Calcul du vecteur de sortie Comme cela est expliqué plus haut (3.59), on obtient simplement les valeurs de sortie du décodeur SISO en faisant la différence des poids stockés (quand ceux-ci existent tout les deux). **Il n'a donc pas été nécessaire de stocker tous les vecteurs obtenus par le Chasing** et on fait donc :

Pour $j = 0 \cdots n$,

Si $PM_j^{(1)} \neq \infty$ et $PM_j^{(0)} \neq \infty$ alors⇒
$r'_j = PM_j^{(0)} - PM_j^{(1)} = P\left(\mathbf{C}^{-1(j)}\right) - P\left(\mathbf{C}^{+1(j)}\right)$

Sinon ⇒
Si $PM_j^{(0)} \neq \infty$ alors ⇒
$r'_j = -\beta$

Sinon ⇒
Si $PM_j^{(1)} \neq \infty$ alors ⇒
$r'_j = +\beta$

On obtient ainsi très simplement la sortie souple du SISO, **sans la boucle supplémentaire de recherche de mot concurrent parmis les mots de code obtenus par Chasing**, contrairement à ce qui est fait dans [Pyn98].

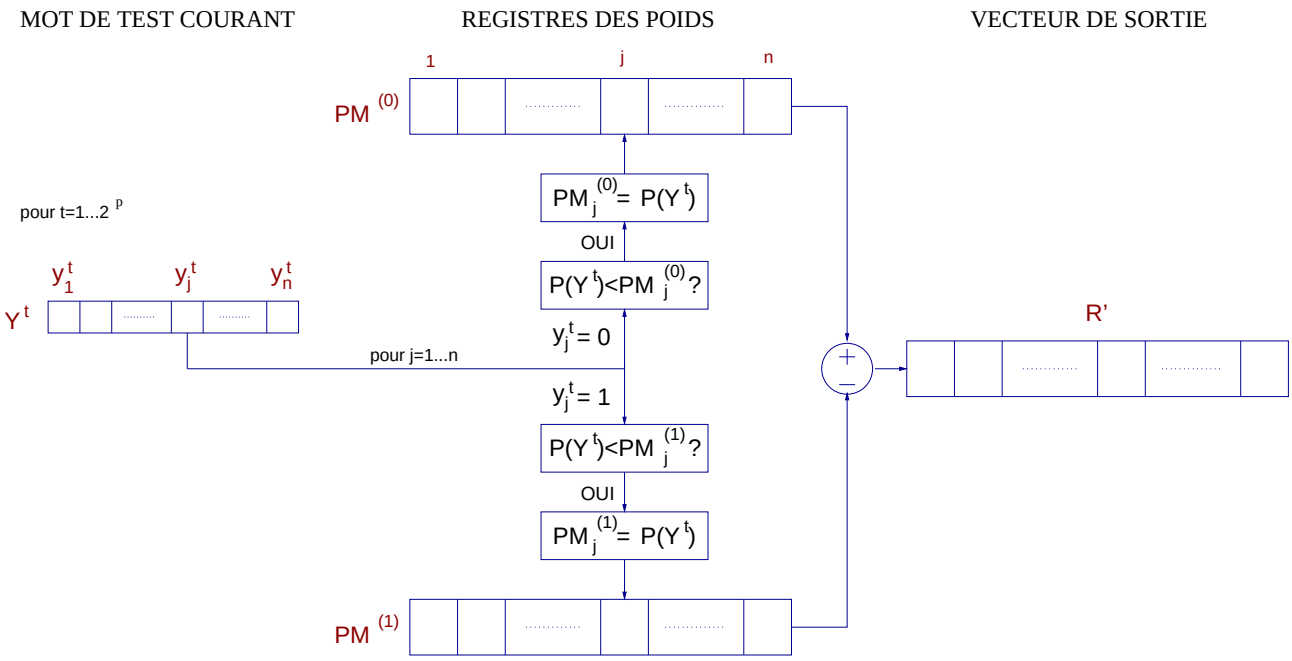


Figure 3.10: Opération de classement des poids

Pour illustrer cet algorithme, nous allons utiliser un exemple simple, décrit par la figure 3.11. Considérons le cas hypothétique où les mots de code comptent 4 bits. On dispose de 4 vecteurs de test décodés :

$\mathbf{Y}^1 = \{0, 0, 0, 1\}$,

$\mathbf{Y}^2 = \{1, 0, 0, 0\}$,

$\mathbf{Y}^3 = \{0, 1, 0, 0\}$,

$\mathbf{Y}^4 = \{1, 1, 0, 0\}$

qui ont respectivement pour poids :

$P(\mathbf{Y}^1) = 0,6$; $P(\mathbf{Y}^2) = 0,7$; $P(\mathbf{Y}^3) = 0,3$ et $P(\mathbf{Y}^4) = 0,4$.

Au départ, les vecteurs de poids sont fixés de manière arbitraire à $+\infty$.

Au premier vecteur reçu $\mathbf{Y}^1$, on stocke son poids aux emplacements correpondant à chaque bit, dans l'un ou l'autre vecteur $PM^{(i)}$ selon la valeur du bit. Les trois premiers bits étant à 0, le poids est stocké dans les trois premières positions du vecteur $PM^{(0)}$. Le dernier bit étant à 1, le poids est stocké dans la dernière position du vecteur $PM^{(1)}$. Ensuite, on considère le vecteur $\mathbf{Y}^2$. Ce vecteur a pour premier bit 1. On compare donc le poids de $\mathbf{Y}^2$ au premier poids du vecteur de poids $PM^{(1)}$. Le poids de valeur 0,7 est inférieur au poids déja stocké, on réaffecte donc cette position à la valeur 0,7. Ensuite, les 3 dernier bits de $\mathbf{Y}^2$ étant à 0, on compare les trois dernières positions de $PM^{(0)}$ au poids de $\mathbf{Y}^2$. Il se trouve être inférieur au poids stocké uniquement dans la dernière position, qui est alors mise à jour, les autres conservant leur valeur de 0,6. On continue avec les vecteurs de test suivants. $\mathbf{Y}^3$ possède un poids inférieur à tout les poids précédemment stockés. La valeur 0,3 est donc affectée aux positions 1, 3 et 4 de $PM^{(0)}$ puisque les bits 1, 3 et 4 de $\mathbf{Y}^3$ sont à 0. La position 2 du vecteur $PM^{(1)}$ est affectée à la valeur 0,3 puisque le bit 2 de $\mathbf{Y}^3$ est à 1. Et enfin le poids du vecteur $\mathbf{Y}^4$ est comparé aux poids précedemment stockés en position 1 et 2 de $PM^{(1)}$ et 3 et 4 de $PM^{(0)}$. Il n'est inférieur au poids précedemment stocké que dans le cas de la position 1 de $PM^{(1)}$, qui est donc affecté à la valeur 0,4.

On obtient ainsi finalement les vecteurs de poids $PM^{(i)}$. Pour connaitre la sortie souple, il suffit de faire la différence entre $PM^{(0)}$ et $PM^{(1)}$ dans les positions où les deux poids sont affectés à une valeur différente de $+\infty$. Sinon, on utilise la valeur β. On obtient donc $\mathbf{R}' = \{-0,1; +0,3; -\beta; -0,3\}$. On remarquera que les signes de ces valeurs correspondent bien aux valeurs des bits de la décision de Chase, c'est-à-dire le vecteur de test de poids le plus faible $\mathbf{Y}^3$.

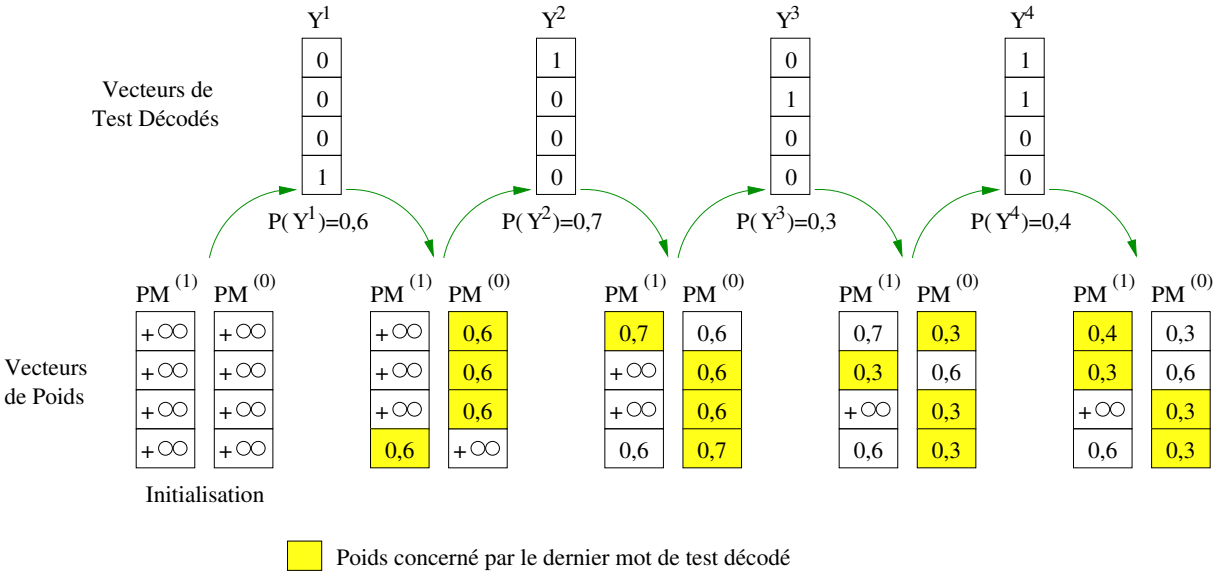


Figure 3.11: Exemple simple illustrant l'algorithme.

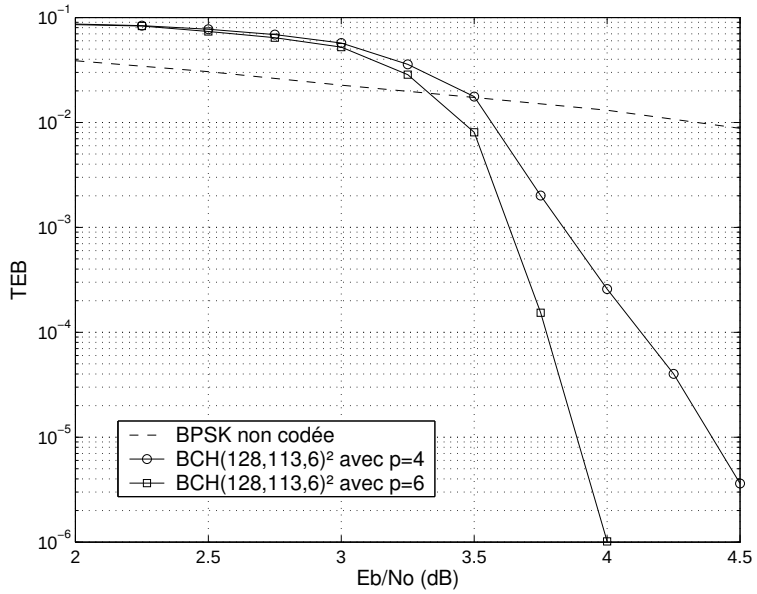


Figure 3.12: Taux d'erreur binaire pour p=4 et p=6 (4 itérations).

La figure 3.12 illustre l'intérêt de l'algorithme. En effet, plus le nombre de vecteurs de tests utilisé est grand, plus cet algorithme sera avantageux. Dans le cas d'un code présentant une capacité de correction élevée, il faut augmenter le nombre de vecteurs tests pour se rapprocher des performances optimales. Dans la figure présentée ici, on considère le cas d'un code BCH$(128,113,6)^2$. Le code BCH utilisé pour construire le code produit a une capacité de correction de 2. On évalue donc ici l'amélioration des

performances apportée par l'augmentation du nombre de vecteurs tests de 2^4 à 2^6. On s'aperçoit que le gain est supérieur à 0,5 dB. Ainsi, la quantité de mémoire nécessaire à notre algorithme étant indépendante du nombre de vecteurs de tests, il est possible d'appliquer cette solution sans pour autant augmenter la quantité de mémoire nécessaire à sa réalisation.

3.5 Conclusion

Dans ce chapitre, nous avons dans un premier temps rappelé les notions classiques du codage canal, et en particulier les codes linéaires en bloc. Nous avons ensuite présenté le principe de fonctionnement de l'algorithme de turbo décodage de R. Pyndiah. Une amélioration structurelle de cette algorithme [GBBV04a] a été proposée. Cette modification permet un allègement important en vue de son intégration, en particulier une importante économie de mémoire, sans en modifier le fonctionnement, ce qui permet d'en faciliter l'intégration dans un système réel.

Chapitre 4

Turbo codes blocs pour le VDSL

Sommaire

4.1 Introduction

L'objectif de ce chapitre est d'exposer une étude réalisée dans le cadre du contrat européen INCA (MEDEA+). Cette étude avait pour but d'évaluer l'interêt des turbo codes dans le cadre des boucles filaires haut débit et en particulier pour le système VDSL, qui représente la prochaine génération de modem après l'ADSL. Ce travail a d'ailleurs donné lieu à deux participations à des conférences [VBGB02][VGBB02]. Cette technologie vise des débits de l'ordre de 50 Mb/s. Une deuxième étude réalisée dans le cadre d'un contrat entre France Telecom et l'ENS Cachan, plus récente, avait pour but d'évaluer les performances du turbo code bloc BCH(128,120,4)^2 en particulier dans le cadre d'un scénario

de bruit prédéfini sur ce canal. Dans un premier temps je présenterai les objectifs de la technologie VDSL, puis détaillerai ensuite le système et décrirai le canal utilisé. Enfin je présenterai les performances des turbo codes blocs dans le cadre d'une telle application.

4.2 Présentation du système VDSL

4.2.1 Objectifs

L'objectif principal de la technologie VDSL et Enhanced VDSL (communément appelé VDSL-2) est de répondre aux attentes de plus en plus exigeantes des opérateurs de télécommunication et des fournisseurs d'accés Internet en terme de débit tout en limitant les investissements et les coûts liés à l'infrastructure. Les investissements et les coûts pour un réseau totalement en fibre optique (FTTH : Fiber To The Home) restent prohibitifs et de nombreux problèmes persistent. C'est pourquoi, une technologie hybride Fibre optique/Cuivre (FTTEx resp. FTTCab : Fiber to the Exchange resp. Cabinet) est la plus appropriée pour les transmissions à haut débit au sein de la boucle locale (réseau téléphonique d'un quartier ou d'une petite ville).

Ainsi les systèmes DSL (ADSL, VDSL et Enhanced VDSL) [Che95][LSWZ96] permettront aux opérateurs de télécommunication de fournir à leurs abonnés, pour un coût modéré, tous les services multimédias en temps réel via les lignes cuivrées du téléphone. L'accés à des débits de plusieurs dizaines de Mbits par seconde permettra sûrement le developpement de la video en direct comme la télévision numérique via le réseau téléphonique, la vidéoconférence et peut-être l'essor du télétravail.

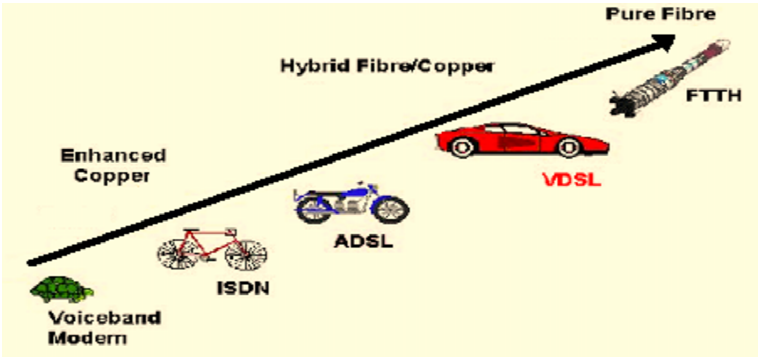


Figure 4.1: Evolution des technologies en terme de débit

4.2.2 Boucle locale actuelle : infrastructure ADSL

La boucle locale de l'opérateur historique est actuellement utilisée par trois différents types d'applications : POTS (téléphonie classique), ISDN (services numériques) et l'ADSL.

L'ouverture à la concurence du marché des communications locales (dégroupage de la boucle locale) a été l'un des moteurs pour le dévelopement de la technologie ADSL

en France. L'infrastructure de cette boucle locale est construite autour d'un Central (Central Office) où toutes les lignes téléphoniques d'un quartier (ou d'une petite ville) et toutes les fibres optiques se raccordant au réseau extérieur convergent. Cette infrastructure est de type FTTEx (Fiber To The Exchange) car les fibres optiques s'arrêtent au Central.

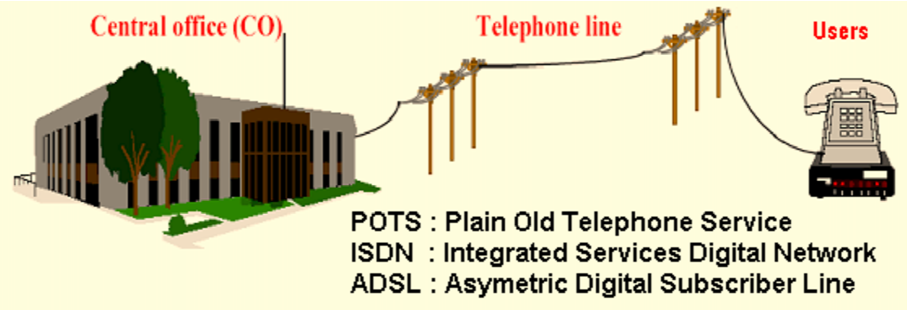


Figure 4.2: Boucle locale ADSL

Pour la technologie ADSL, la longueur maximale des lignes cuivrées téléphoniques est comprise entre 4 et 6 km pour avoir un débit suffisant. C'est pourquoi le developpement rural de cette technologie a été plus difficile et plus coûteux.

4.2.3 Infrastructure VDSL

Contrairement à l'ADSL, la technologie VDSL nécessite une modification de l'infrastructure de la boucle locale afin de pouvoir fournir des débits beaucoup plus importants à tous les abonnés. Ainsi pour utiliser le VDSL, le réseau en fibre optique devra être étendu, au-delà du Central, jusqu'aux zones résidentielles ou industrielles, où un ONU (Optical Network Unit) fera la liaison entre la fibre optique et les paires cuivrées reliant les différents abonnés. Dans le cas d'une zone résidentielle par exemple (Type de structure FTTCab : Fiber To The Cabinet), l'ONU sera placé dans un local technique le long du trottoir.

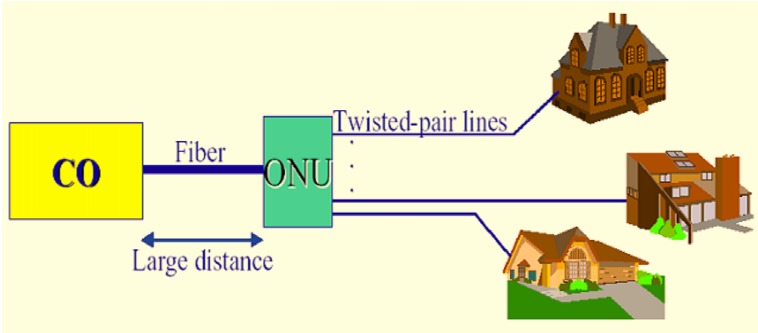


Figure 4.3: Structure d'une connexion VDSL

Autre exemple, dans le cadre d'une entreprise ou d'un immeuble, si on place une ONU à la base du bâtiment (FTTB : Fiber To The Building), des liaisons VDSL à trés haut débit pourront parcourir tous les étages. Cependant, la structure FTTEx, c'est-à-dire sans ONU, déja utilisée pour l'ADSL est aussi envisagée dans la norme VDSL mais avec des distances inférieures à 1.5 km pour les lignes téléphoniques issues directement du Central. Cette structure FTTEx sera donc utilisée pour le VDSL uniquement pour les abonnés se situant au voisinage du Central (CO).

4.2.4 Bandes de fréquences

Grâce aux nouvelles infrastructures utilisées par la technologie VDSL (où les longueurs de fils cuivrés sont limitées à 1.5 km), il est possible alors d'utiliser une large bande de fréquences afin d'obtenir des débits très élevés malgré une atténuation importante en fonction de la longueur de la ligne.

La bande de fréquences du VDSL prévue à l'origine s'étendait de 138 kHz (fin de la bandes ISDN) jusqu'à 30 MHz [ETS03a][ETS03b] afin de ne pas perturber les anciennes applications (POTS et ISDN).

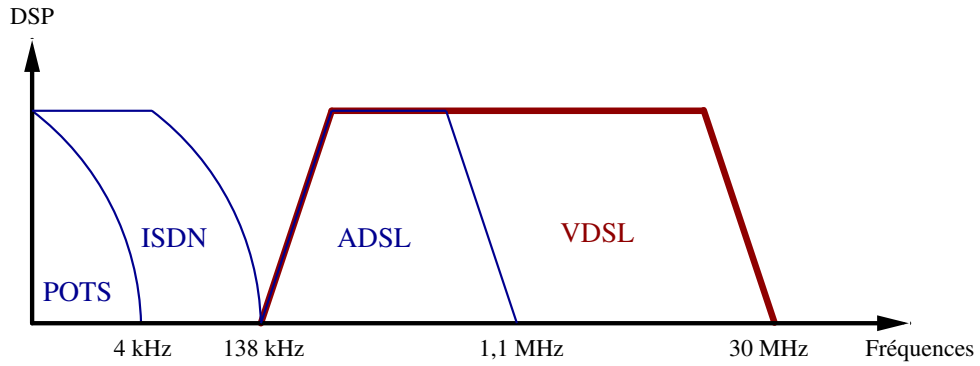
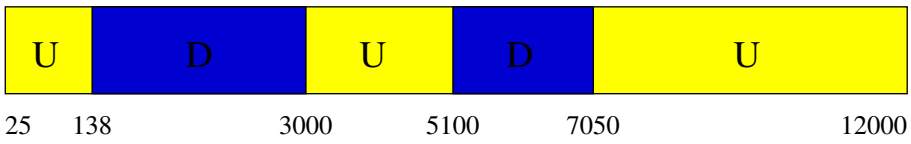
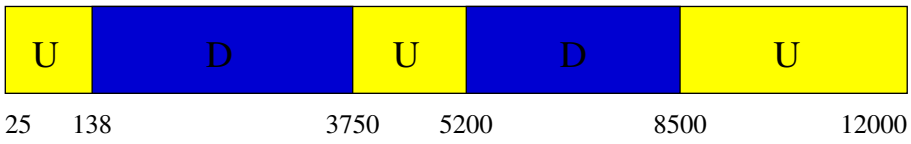


Figure 4.4: Bandes de fréquences VDSL

C'est l'effet de peau qui est responsable de la limitation en fréquence de ce canal de transmission et même si la gamme de fréquence du VDSL s'étend en théorie jusqu'à 30 MHz, dans les dernières normes ETSI concernant la technologie VDSL, la fréquence maximale utilisable se limite à 12 MHz car, au-delà, l'atténuation est si importante que des résultats satisfaisants se limitent à une dizaine de mètres de ligne.



Plan code 997 – Débits symétriques



Plan code 998 – Débits asymétriques

Figure 4.5: Plan de fréquences pour le VDSL.

Cette bande de fréquence VDSL comporte 2750 porteuses allant de 138 kHz à 12 MHz chacune espacée de $\Delta f = 4.3125$ kHz. Cet écart fréquentiel Δf entre les porteuses est le même que pour l'ADSL qui utilise les 256 plus basses fréquences de cette bande (<1.1 MHz). L'ensemble de ces porteuses est divisé en 5 sous-bandes : 2 bandes sont réservées pour le débit montant (upload : du terminal vers le réseau), 2 pour le débit descendant (download) et une dernière bande optionnelle utilisable en upload (de 25 kHz à 138 kHz c'est-à-dire l'ajout de 26 porteuses).

De plus, il existe dans la norme VDSL deux plans distincts de fréquences utilisables par les opérateurs pour obtenir des débits symétriques (Plan code 997) ou asymétrique (Plan code 998) favorisant le sens descendant pour des applications de type diffusion de TV numérique.

4.2.5 Comparaison entre les débits ADSL/VDSL

Le principal avantage du VDSL est qu'il permettra d'obtenir un débit total (upload + download) allant jusqu'à 56 Mb/s. Mais ce débit trés élevé de 56 Mb/s n'est atteint que pour les distances inférieures à 300 m et la figure 4.6 compare les débits et la portée (distance) entre l'ADSL et le VDSL.

On observe donc que le déploiement de la technologie VDSL ne pourra pas dépasser des distances supérieures à 1.5 km. Par contre, le système ADSL permet d'obtenir des débits inférieurs mais sur une plus grande distance jusqu'a 6 km. Ces deux technologies ne sont donc pas réellement en concurrence, le VDSL proposant des débits beaucoup plus élevés sur de courtes distances.

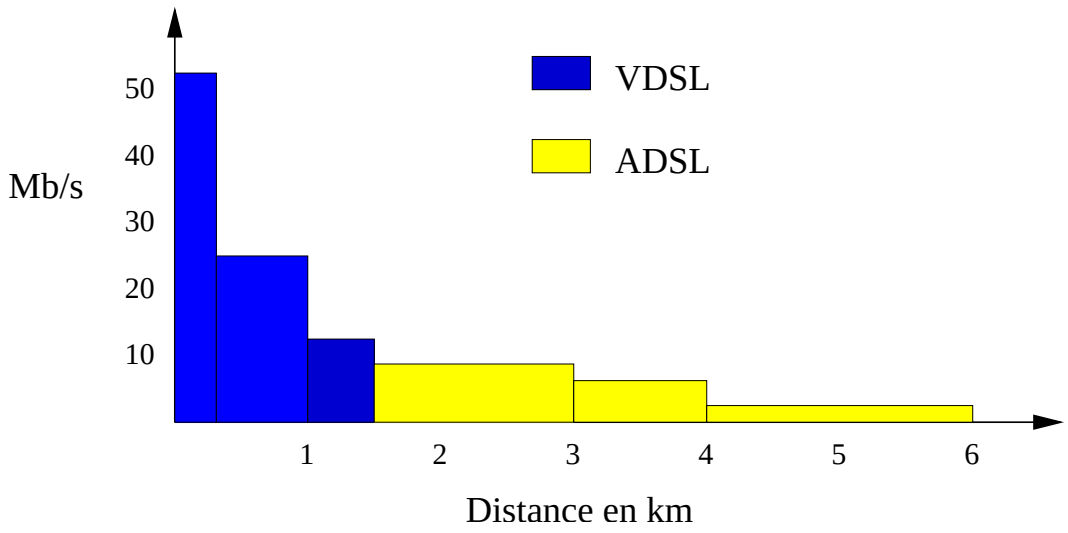


Figure 4.6: Comparaison des débits VDSL/ADSL.

4.2.6 Autres éléments de la norme VDSL

Le système VDSL utilise une modulation DMT. Chaque porteuse transporte donc un symbole MAQ dont la taille peut varier entre 2 points (1 bit par symbole) et 32768 points (15 bits par symbole). La taille du symbole utilisé pour chaque porteuse dépend du SNR disponible pour cette porteuse et on utilise donc le procédé de bitloading pour s'adapter au mieux aux caractéristiques du canal.

Un symbole DMT est composé de la somme de 2750 porteuses. Il présente un suffixe et un préfixe cyclique. La durée totale d'un symbole est de 250 μs et on emet donc 4000 symboles par seconde.

La première génération du système VDSL utilisera pour le codage canal le même schéma que l'ADSL, la modulation codée en treillis concaténée à un code de typoe Reed-Solomon.

4.3 Modélisation du canal VDSL

L'objectif de cette étude était d'évaluer les débits envisageables à travers le canal de transmission du VDSL. C'est pourquoi, il a fallu modéliser et programmer le canal de transmissions en respectant la dernière proposition de norme ETSI [ETS03a][ETS03b] concernant les transmission VDSL.

4.3.1 Fonction de transfert des câbles

Dans la norme VDSL, il existe deux types de câbles : le TP100 est très utilisé pour les transmissions enterrées et le TP150 possède la même quantité de cuivre (diamètre 0.5 mm) mais sa partie isolante est plus performante pour permettre l'assemblage de

plusieurs paires cuivrées dans la même gaine. A l'aide des constantes (4.1) des câbles TP100 et TP150 données par la norme VDSL, on définit les paramètres de ces cables [Com96] : impédance série linéique Z_{s0} et admittance parallèle linéique Y_{s0}. On définit l'impédance normalisée $R_v = 135\,\Omega$ en bout de ligne. Les caractéristiques d'un cable pour une longueur donnée L sont donc :

$$\begin{aligned} Z_s &= L \times Z_{s0} \\ Y_p &= L \times Y_{p0} \\ \gamma &= \sqrt{Z_s \times Y_p} \\ Z_0 &= \sqrt{Z_s / Y_p} \end{aligned} \tag{4.1}$$

où Z_s est l'impédance série, Y_p l'admittance parallèle,γ est l'exposant linéique de propagation et Z_0 l'impédance caractéristique.

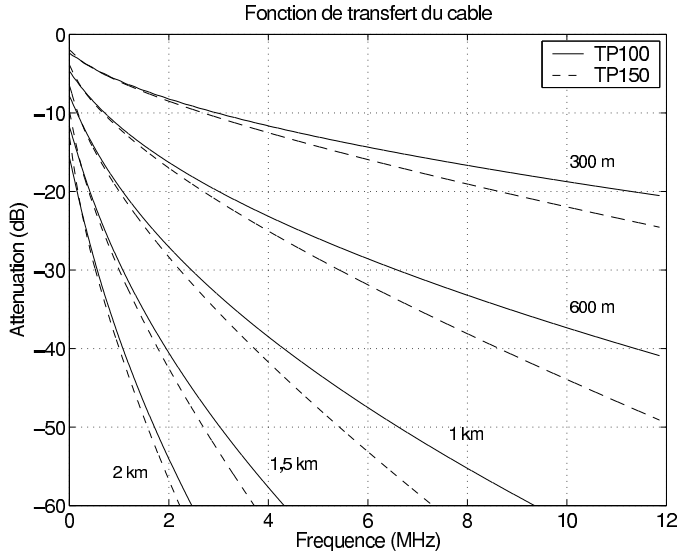


Figure 4.7: Fonction de transfert de la ligne pour différentes longueurs.

On peut également définir la résistance, l'inductance, la capacitance et la conductance :

$$\begin{aligned} R_s &= \mathrm{Re}\,(Z_s) \\ L_s &= \mathrm{Im}\,(Z_s/\omega) \\ C_p &= \mathrm{Im}\,(Y_p/\omega) \\ G_p &= \mathrm{Re}\,(Y_p) \end{aligned}$$

Les paramètres S du cable sont alors donnés par :

$$\mathbf{S} = \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix} = \frac{1}{\left(\frac{Z_0}{R_v} + \frac{R_v}{Z_0}\right)\tanh(\gamma) + 2} \begin{bmatrix} \left(\frac{Z_0}{R_v} - \frac{R_v}{Z_0}\right)\tanh(\gamma) & \frac{2}{\cosh(\gamma)} \\ \frac{2}{\cosh(\gamma)} & \left(\frac{Z_0}{R_v} - \frac{R_v}{Z_0}\right)\tanh(\gamma) \end{bmatrix}$$

s_{21} et s_{12} sont les paramètres de transmission, s_{11} et s_{22} sont les paramètres de reflection. La fonction de transfert H du cable est donnée par s_{21} :

$$H(f,L) = \frac{2}{\left(\left(\frac{Z_0}{R_v} + \frac{R_v}{Z_0}\right)\tanh(\gamma) + 2\right)\cosh(\gamma)} \tag{4.2}$$

La figure 4.7 illustre l'atténuation obtenue avec les cables TP100 et TP150 pour plusieurs longueurs de lignes en fonction de la fréquence.

4.3.2 Modélisation en fréquence du signal transmis, des interférences et du bruit

4.3.2.1 Modelisation du signal transmis

La norme du VDSL définit les caractéristiques des signaux transmis du coté utilisateur (NT : Network Termination) pour le signal montant (Upload) et du côté des serveurs (LT : Line Termination) pour le signal descendant (Download) :

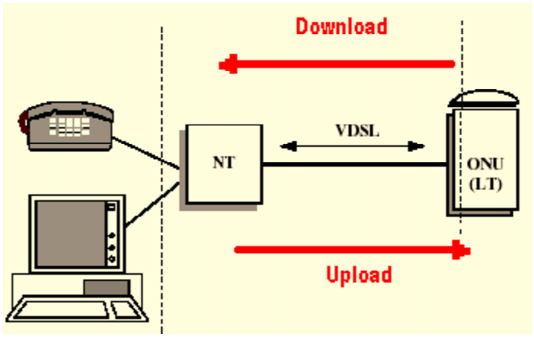


Figure 4.8: NT, LT, download, upload.

Comme cela a été présenté dans le paragraphe 4.2.4, les bandes de fréquences pour les signaux montants et descendants (fig. 4.5) imposent des spectres d'émission différents selon la position du modem (NT ou LT). Les figures suivantes illustrent la densité spectrale de puissance émise du plan 997 symétrique en fonction de la fréquence dans les deux cas :

- Pour le signal montant "Upload" (NT => LT), le masque de spectre est a peu prés constant sur l'ensemble des porteuses en upload (figure 4.9).
- Pour le signal descendant "Download" (LT => NT), le masque de spectre, dans le cas où l'ONU (modem LT) se trouve dans la position FTTEx (Exchange), est donné par la figure 4.10.

Dans le cas où l'ONU se trouve dans la position FTTCab (Cabinet), la puissance émise dans la bande de fréquences utilisée par l'ADSL (<1.1 MHz) sera limitée. Ce scenario

tient donc compte de la compatibilité avec le système ADSL préexistant et le débit en download du système VDSL sera donc diminué.

Dans le cadre de la norme VDSL, ces spectres respectent le niveau maximum autorisé de puissance totale émise qui est de 14,5 dBm pour le signal descendant et 11,5 dBm pour le signal montant. Ces puissances totales émises varient en fonction du plan de fréquence (997 ou 998) et du scenario de déploiement (FTTEx ou FTTCab).

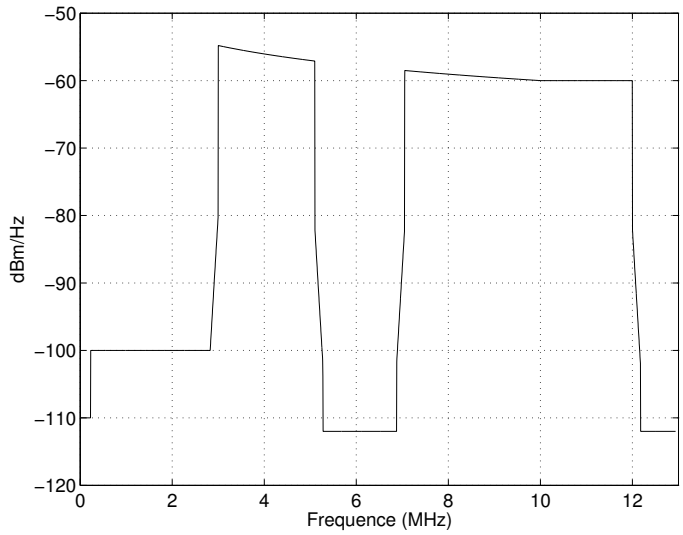


Figure 4.9: Spectre d'émission pour le signal montant (upload).

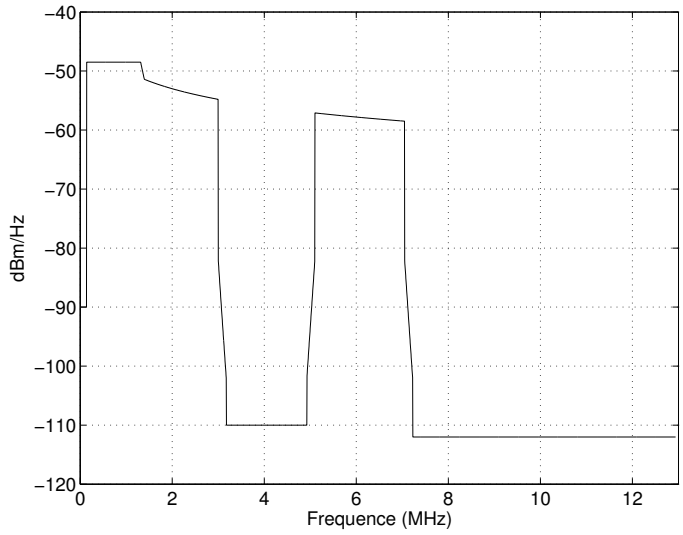


Figure 4.10: Spectre d'émission pour le signal descendant (download) dans le cas FTTEx.

4.3.2.2 Modélisation des interférences et des bruits

Le canal de transmission VDSL est fortement perturbé par plusieurs sources de bruits et d'interférences :

- Le bruit blanc gaussien additif (AWGN : Additive White Gaussian Noise). Dans la norme ETSI du VDSL le niveau de l'AWGN est de -140 dBm/Hz,
- Les phénomènes de diaphonies entre les lignes VDSL,
- Les interférences dues aux autres systèmes utilisant le même support de transmission.

Il existe deux types de diaphonies :

- Le NEXT (Near-End Crosstalk) ou paradiaphonie, où l'emetteur (Tx) perturbe les récepteurs (Rx) qui lui sont proches. Ainsi dans le cas de l'utilisateur (position du modem NT), le signal descendant reçu est perturbé par le signal montant émis et dans le cas de l'opérateur (position LT), c'est le signal montant reçu qui est perturbé par le signal descendant émis.
- Le FEXT (Far-end crosstalk) où télédiaphonie, modélise l'interférence due à un émetteur (Tx) sur les récepteurs (Rx) qui sont à l'autre extrémité de la ligne. Ainsi dans le cas de l'utilisateur (position NT), le signal descendant reçu est perturbé par des signaux descendants dans la même direction mais venant d'autres émetteurs distants et dans le cas de l'opérateur (position LT), c'est le signal montant reçu qui est perturbé par d'autres signaux montants qui viennent de plusieurs utilisateur différents.

Ainsi l'influence du FEXT et du NEXT dépend directement des spectres d'émission que l'on a défini au paragraphe 4.3.2.1. La puissance réelle de ces diaphonies est donc obtenue par filtrage des signaux émis avec deux filtres [ETS03a] modélisés par les fonctions de transfert (4.3) et (4.4).

$$\text{NEXT} : \quad H_1(f,L) = K_{xn}\,(f/f_0)^{0.75}\sqrt{1-|H(f,L)|^4} \tag{4.3}$$

$$\text{FEXT} : \quad H_2(f,L) = K_{xf}\,(f/f_0)\sqrt{L/L_0}\,|H(f,L)| \tag{4.4}$$

avec $K_{xn} = 10^{-50/20}$, $K_{xf} = 10^{-45/20}$, $f_0 = 1\,\text{MHz}$, $L_0 = 1\,\text{km}$ et H représente la fonction de transfert du cable défini au paragraphe 4.3.1.
Bien que les interférences NEXT correspondent à des puissances plus élevées que les interférences FEXT, le fait qu'elles s'appliquent sur des bandes de fréquences disjointes limite leur rôle perturbateur aux extrémités des sous bandes. Enfin, le bruit extérieur, ou "Alien Noise", est caractérisé par de fortes perturbations en basse fréquences (<1.1 MHz) et il ne dépend que très peu de la position du modem (NT ou LT). Sa modélisation, et donc son influence, dépend du nombre d'éléments perturbateurs qui se situent au sein du câble de distribution.

4.3.2.3 Calcul du RSB total et algorithme de "Bitloading"

Connaissant les puissances émises et les sources de bruits, il est possible de calculer le Rapport Signal à Bruit (RSB) à la réception en tenant compte de l'atténuation du signal émis et de toutes les perturbations. Sachant que les effets des interférences (FEXT, NEXT et Alien Noise) sont différents selon la position du modem (utilisateur et opérateur), il est nécessaire de calculer deux RSB : l'un pour le signal montant et l'autre pour le signal descendant. Par exemple, dans le cas du plan de fréquence symétrique 997 (cf. figure 4.5), le RSB en liaison descendante (côté utilisateur) est calculé entre 138 et 3000 kHz et entre 5100 et 7050 kHz. Pour la liaison montante (côté opérateur), le RSB est calculé entre 3000 et 5100 kHz et 7050 et 12000 kHz. Pour calculer le RSB total résultant de l'ensemble des perturbations, il faut :

- Mesurer la densité de puissance du signal reçu (en W/Hz) en filtrant la puissance du signal émis par la fonction de transfert H du câble pour simuler l'atténuation due à la ligne.

- Mesurer la densité de puissance totale des bruits (AWGN+FEXT+NEXT+Alien) avec la formule 4.5:

$$P_b = \left(\sum_{i \in NEXT, FEXT, Alien, AWGN} P_{b_i}^{1/0.6} \right)^{0.6} \tag{4.5}$$

où P_b est la densité spectrale totale de bruit (W/Hz) et P_{b_i} est la densité spectrale de chaque bruit. Par exemple, dans le cas où il y a uniquement N interféreurs de type NEXT, (4.5) s'écrit $P_b = N^{0.6} P_{bNEXT}$.

- Enfin, on calcule le rapport de ces deux densités spectrales de puissance pour obtenir le RSB en dB de chaque porteuse j :

$$RSB_j = 10 \log_{10} \left(\frac{P_{signal_j}}{P_{bruit_j}} \right) - 8\,\text{dB} \tag{4.6}$$

La soustraction de 8 dB correspond à la perte de RSB due à l'implantation physique et intègre ce que l'on appelle la marge de bruit (SNR margin) de 6 dB.

La figure 4.11 illustre un exemple de RSB sur la bande de fréquence VDSL (138 - 12000 kHz) pour une longueur de ligne de 600 m. On observe des chutes de RSB dûes aux NEXT à l'intersection des bandes de fréquences utilisées en upload et en download. Par conséquent, les porteuses qui se situent au bord des bandes sont inutilisables.

A partir de la connaissance du RSB pour chaque porteuse, le nombre de bits transportés par chaque porteuse est déterminé par un algorithme de bitloading. Le bitloading consiste à determiner le nombre de bits maximum que l'on peut allouer à une porteuse sans dépasser un taux d'erreur binaire cible ($TEB < 10^{-7}$ dans le cas du VDSL) pour la porteuse considérée.

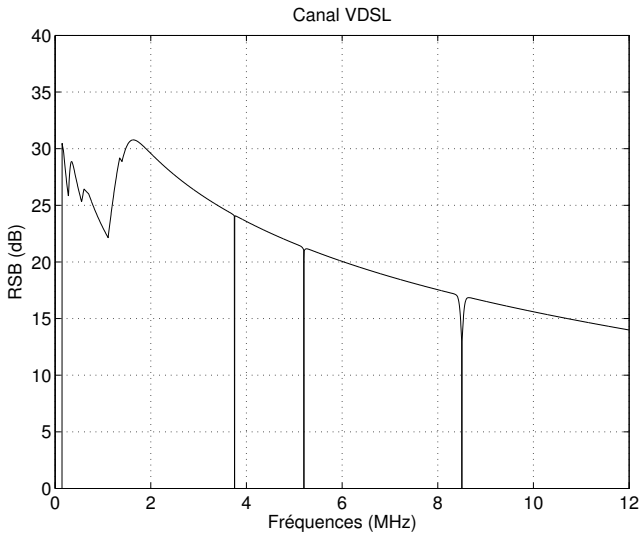


Figure 4.11: Rapport Signal à Bruit pour une distance de 600 m sur cable TP100.

Il suffit pour cela de comparer la courbe de TEB d'une MAQ donnée au RSB disponible sur la porteuse. Il faut choisir le nombre de bit maximum tel que, pour la modulation correspondante à ce nombre de bit, la valeur de RSB donnant le taux d'erreur cible soit inférieur au RSB effectivement disponible sur cette porteuse. Cette valeur est bien entendu influencée par l'utilisation d'un code correcteur d'erreur. C'est pourquoi dans la partie suivante nous allons étudier l'influence du codage sur les débits obtenus.

4.4 Gains de codage et débits

Afin de pouvoir calculer les débits pour différents codes et différents scénarii prévus au cahier des charges, il est donc nécessaire de connaitre tous les gains de codage (pour des symboles MAQ de 1 à 15 bits) pour appliquer l'algorithme de bitloading. Nous comparerons ici les débits obtenus avec les turbo codes à ceux obtenus par l'utilisation d'un schema de codage classique, qui est celui utilisé dans l'ADSL, consistant en une modulation codée en treillis optionnelle concaténée avec un code RS(144,128). Ainsi dans la suite, nous allons expliciter les résultats obtenus en terme de gains de codage pour ces deux types de codes.

4.4.1 Turbo Codes Blocs

Les gains de codage des turbo codes ont été calculé pour un taux d'erreur cible de 10^{-7}. Les figures 4.12 à 4.14 donne le résultat des courbes de TEB pour un code BCH(128,120,4)^2 en fonction de E_b/N_0. Les gains de codage sont résumés dans le tableau 4.1 récapitulatif des gains.

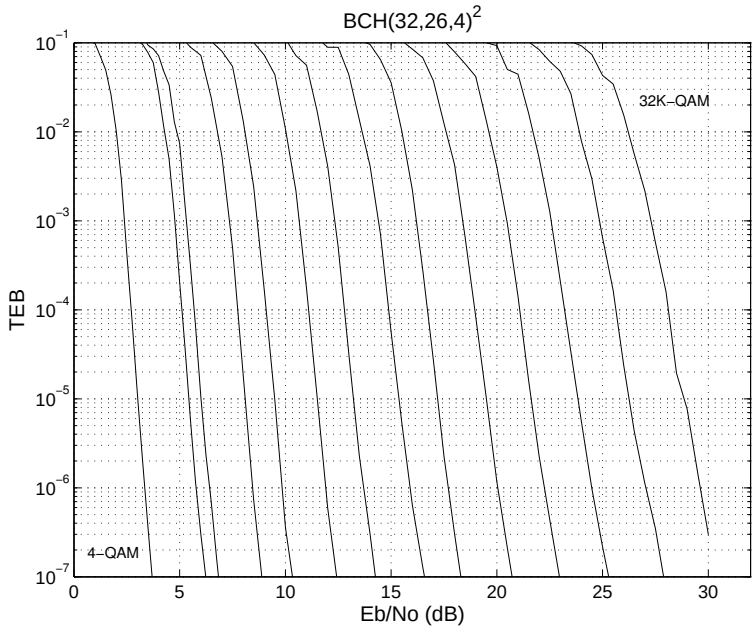


Figure 4.12: Courbes de TEB pour un turbo code bloc BCH(32,26,4)^2.

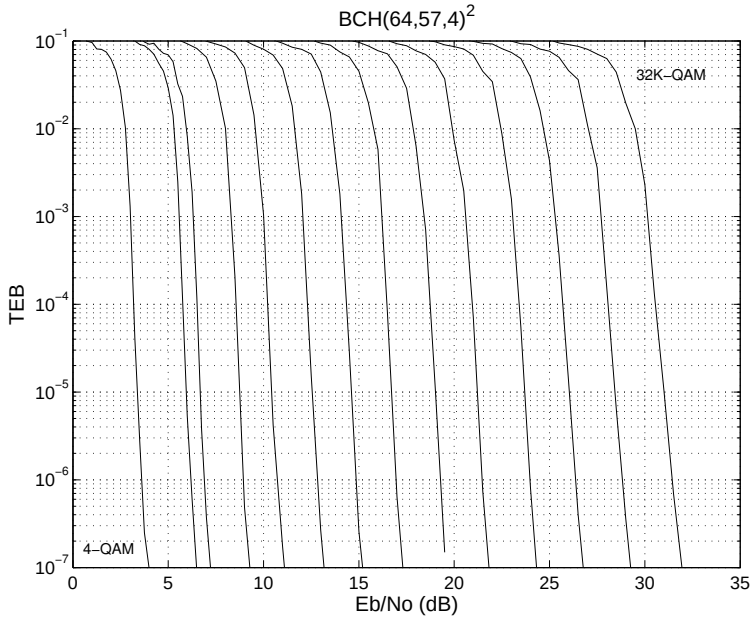


Figure 4.13: Courbes de TEB pour un turbo code bloc BCH(64,57,4)^2.

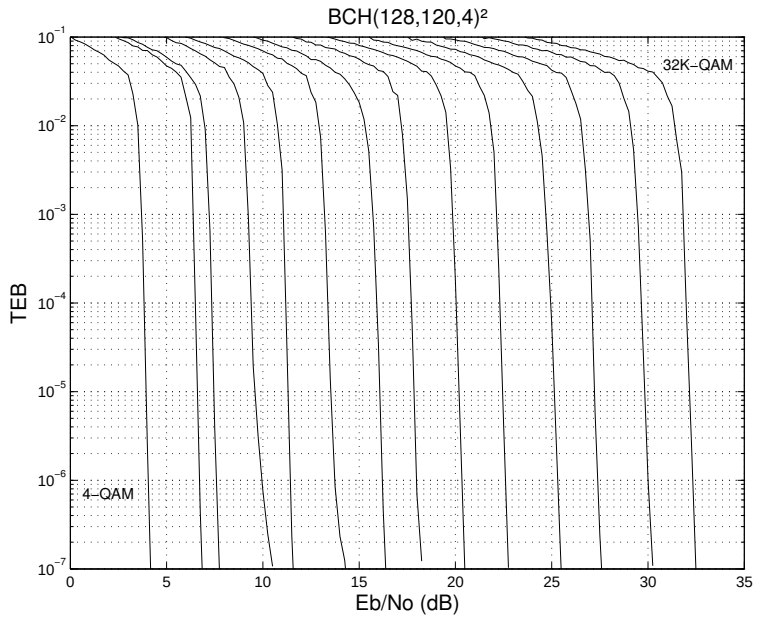


Figure 4.14: Courbes de TEB pour un turbo code bloc BCH(128,120,4)^2.

Taille QAM	Gain TCM+RS	Gain BCH(32,26,4)^2	Gain BCH(64,57,4)^2	Gain BCH(128,120,4)^2
2	4,5	7,8	7,5	7,1
4	6,8	7,8	7,5	7,1
8	6,8	8,0	8,0	7,5
16	6,2	8,3	7,5	7,5
32	5,8	8,3	8,0	6,7
64	6,0	9,2	8,5	8,0
128	5,9	9, 5	9,0	7,6
256	5,8	10,0	9,2	8,1
512	5,8	10,0	9,5	8,4
1024	5,8	10,5	9,8	8,8
2048	5,7	10,5	10,0	9,0
4096	5,6	11,2	10,0	9,0
8192	5,6	11,8	10,4	9,5
16384	5,7	11,8	10,8	9,6
32768	5,9	12,2	11,0	9,8

Table 4.1: Tableau récapitulatif des gains de codage.

4.4.2 Modulations codées en treillis

La modulation codée en treillis (Treillis Coded Modulation : TCM) est une technique qui effectue à la fois la modulation et le codage canal [Ung82][Wei84a][Wei84b]. En utilisant un étiquetage particulier des bits, on peut faire apparaitre des différences de "protection" entre les différents bits du symbole. Ainsi, seuls les bits les moins protégés sont codés par cette technique, de façon a ce que la distance entre les symboles codés soit maximale tout en conservant un très bon rendement. Ce schéma de modulation est actuellement utilisé en concaténation avec un code de Reed Solomon dans la norme ADSL/VDSL.

Taille QAM	Rendement TCM+RS
2	0,889
4	0,666
8	0,740
16	0,777
32	0,800
64	0,815
128	0,825
256	0,833
512	0,839
1024	0,844
2048	0,848
4096	0,852
8192	0,855
16384	0,857
32768	0,859

Table 4.2: Tableau récapitulatif du rendement de la modulation codée.

$BCH(32,26,4)^2$	$BCH(64,57,4)^2$	$BCH(128,120,4)^2$
0,660	0,793	0,879

Table 4.3: Tableau récapitulatif des rendements de turbo codes blocs.

Comme on peut l'observer sur la figure 4.15, les modulations codée en treillis présentent une pente beaucoup moins forte que les turbo codes blocs. Ce comportement se ressent particulièrement quand la taille de la constellation augmente. Les resultats en terme de gain de codage sont résumés dans le tableau 4.1.

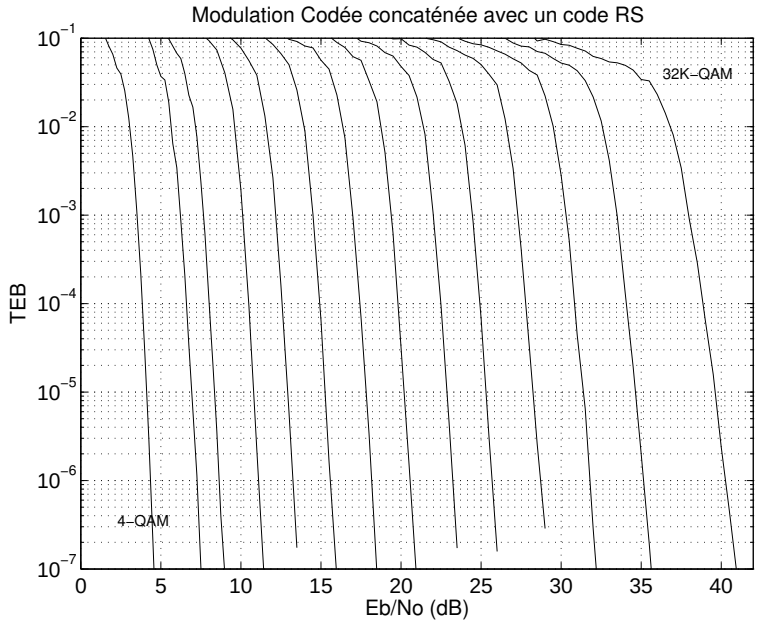


Figure 4.15: Courbes de TEB pour une modulation codée concaténée avec un code RS.

4.4.3 Calculs de débits

Un premier paragraphe présentera les profils définis dans la norme VDSL. Je presenterai ensuite des comparatifs de courbes de débit en fonction de la longueur de lignes pour les différents schémas de codage envisagés. Il sera ainsi possible de déterminer des gains en terme de distance pour obtenir un profil de débit donné.

4.4.3.1 Profils de débits

La norme VDSL définit des profils de débits (cf fig. 4.4) : les profils S1 à S5 sont les profils utilisant le plan de fréquence symétrique, les profils A1 à A4 utilisent le plan de fréquence asymétrique. Le but est bien évidemment d'obtenir un profil donné sur la longueur de ligne la plus longue possible, afin d'optimiser le service fourni au client. On espère donc obtenir des gains non négligeables en utilisant un schéma de codage plus sophistiqué que l'actuelle concaténation du code RS avec la modulation codée présente dans la norme.

Pour chaque profil la portée sera la distance la plus grande possible permettant d'atteindre les deux débits (download et upload). Le système VDSL atteint ses limites pour environ 1.5 km comme nous l'avons vu dans le paragraphe 4.2.5. C'est pourquoi le débit utile est calculé pour des longueurs de lignes allant de 50 à 2000 m pour chaque scénario de simulations. De nombreux paramètres (Puissance du signal, caractéristiques du bruit,...) sont variables afin d'étudier les débits utiles sous tous les scénarii possibles que nous développons ci-aprés.

- 2 types de cables sont utilisés : le TP100 et TP150. Le choix du cable modifie la fonction de transfert (cf equation 4.7) de la ligne et donc influe sur l'atténuation du signal émis.
- 2 plans de fréquences sont envisagés dans la norme VDSL : le plan code 997 qui donne des débits symétriques (Download = Upload) et le plan code 998 correspond à des débits asymétriques (Download > Upload). Ces bandes de fréquences sont définies figure 4.5.
- 2 techniques de deploiement du système VDSL. Premièrement, le cas où l'ONU se trouve dans le central (Cas FTTEx), puis le cas où l'ONU est installée sur un trottoir ou un sous-sol d'immeuble (FTTCab). Le cas FTTCab ne permet pas d'utiliser les basses fréquences utilisées par la technologie ADSL (<1.1 MHz). Les conséquences sont néfastes uniquement sur le débit montant.
- Nombre d'éléments perturbateurs FEXT et NEXT (cf. eq.4.3 et 4.4). Ce nombre correspond en réalité au nombre de lignes VDSL se situant dans le même cable de distribution. Dans le cadre de nos simulations, ce nombre varie de 0 (pas de diaphonie) à 49.
- Choix du scénario pour "l'alien noise". Dans le cas de nos simulations, on s'est concentré sur deux scénarios de bruits définis dans la norme VDSL sous la désignation "B" et "E". Ces deux scénarios représentent un nombre d'éléments perturbateurs (nombre de lignes ADSL, ISDN ou POTS) médian. Le scénario de bruit "E" est défini dans le cadre du déploiement depuis le Central (FTTEx) alors que le scenario de bruit "B" est défini dans le cadre du déploiement depuis le Cabinet (FTTCab).

PROFILS	Download (Kbit/s)	Upload (Kbit/s)
S1	6400	6400
S2	8576	8576
S3	14464	14464
S4	23168	23168
S5	28288	28288
A1	6400	2048
A2	8576	2048
A3	14464	3072
A4	23168	4096

Table 4.4: Profils de débits du VDSL.

4.4.3.2 Scénarios et débits calculés

Nous allons ici considérer plusieurs scénarios d'implantation du VDSL et les débits obtenus avec différents schémas de codage canal. Nous définirons dans chaque cas les conditions de simulations.

Scénario 1 :

Type de câble :	TP150
Plan de fréquence :	998
Scénario de déploiement :	FTTEx
Nombre de FEXT-NEXT :	4
Alien Noise :	Scénario “E”

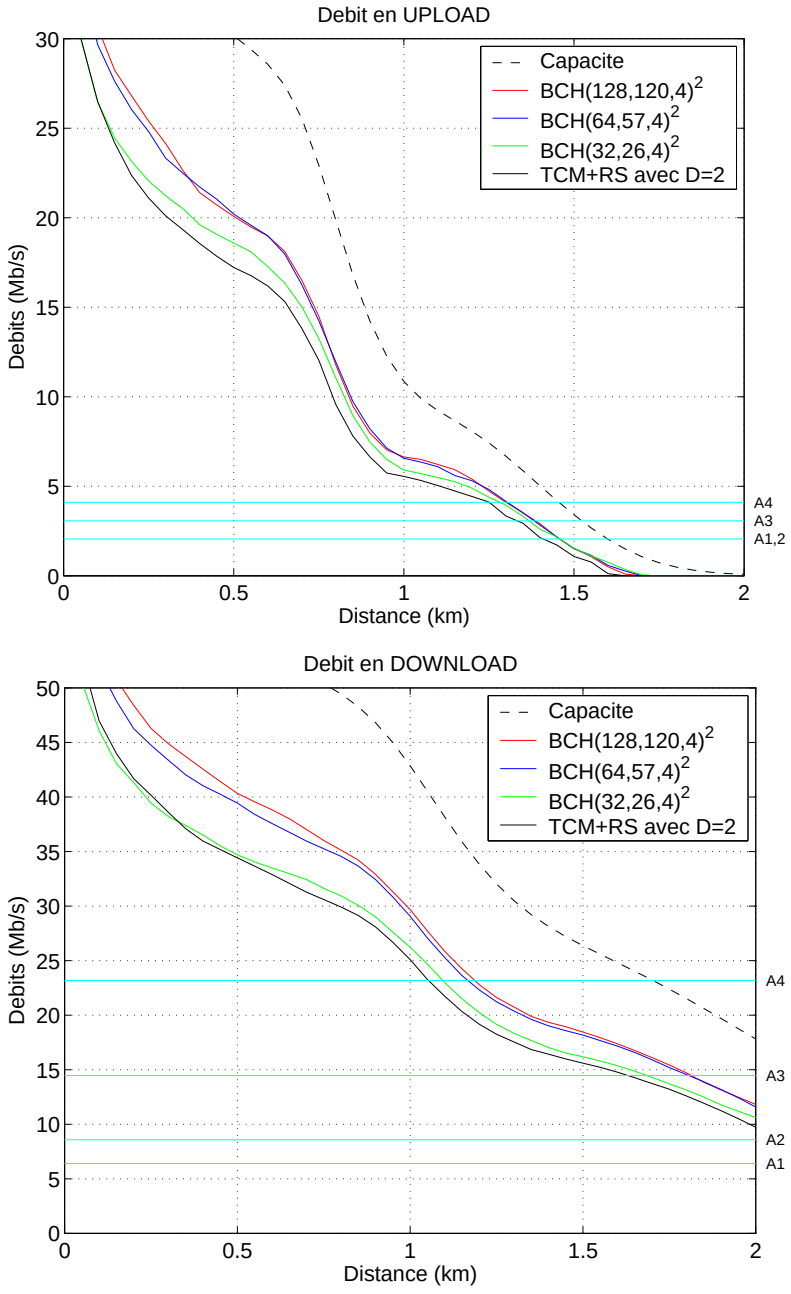


Figure 4.16: Courbe de débit en fonction de la longueur de ligne.

Scénario 2 :

Type de câble :	TP150
Plan de fréquence :	997
Scénario de déploiement :	FTTEx
Nombre de FEXT-NEXT :	4
Alien Noise :	Scénario “E”

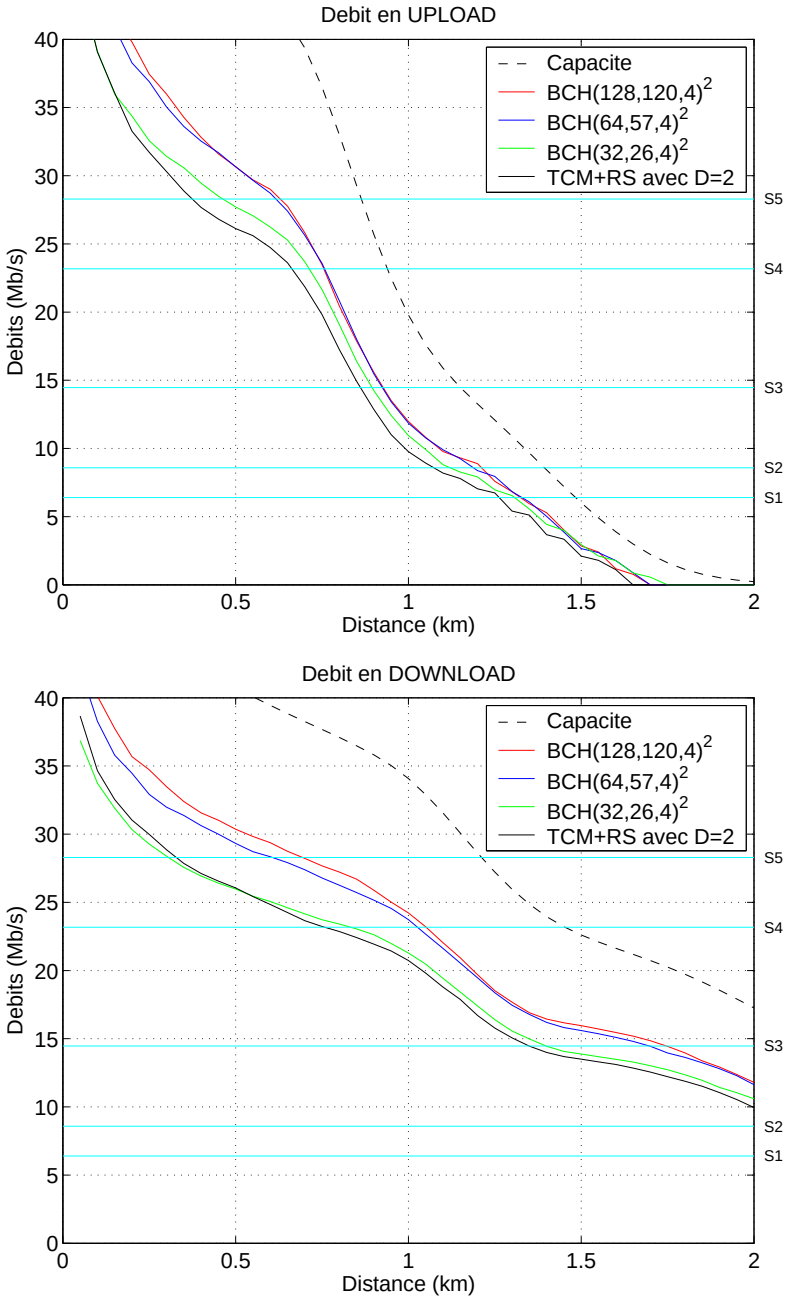


Figure 4.17: Courbe de débit en fonction de la longueur de ligne.

Scénario 3 :

Type de câble :	TP100
Plan de fréquence :	998
Scénario de déploiement :	FTTEx
Nombre de FEXT-NEXT :	4
Alien Noise :	Scénario “E”

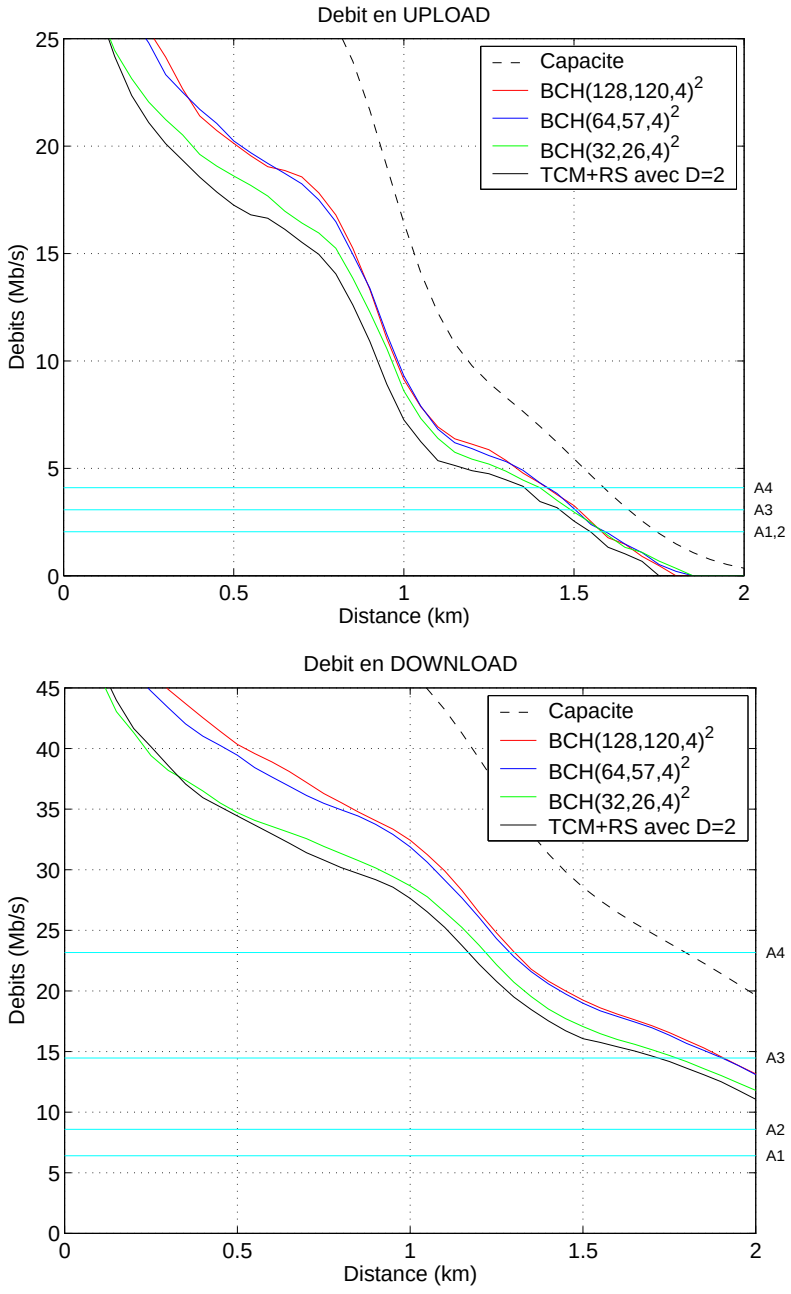


Figure 4.18: Courbe de débit en fonction de la longueur de ligne.

Scénario 4 :

Type de câble :	TP150
Plan de fréquence :	998
Scénario de déploiement :	FTTEx
Nombre de FEXT-NEXT :	49
Alien Noise :	Scénario “E”

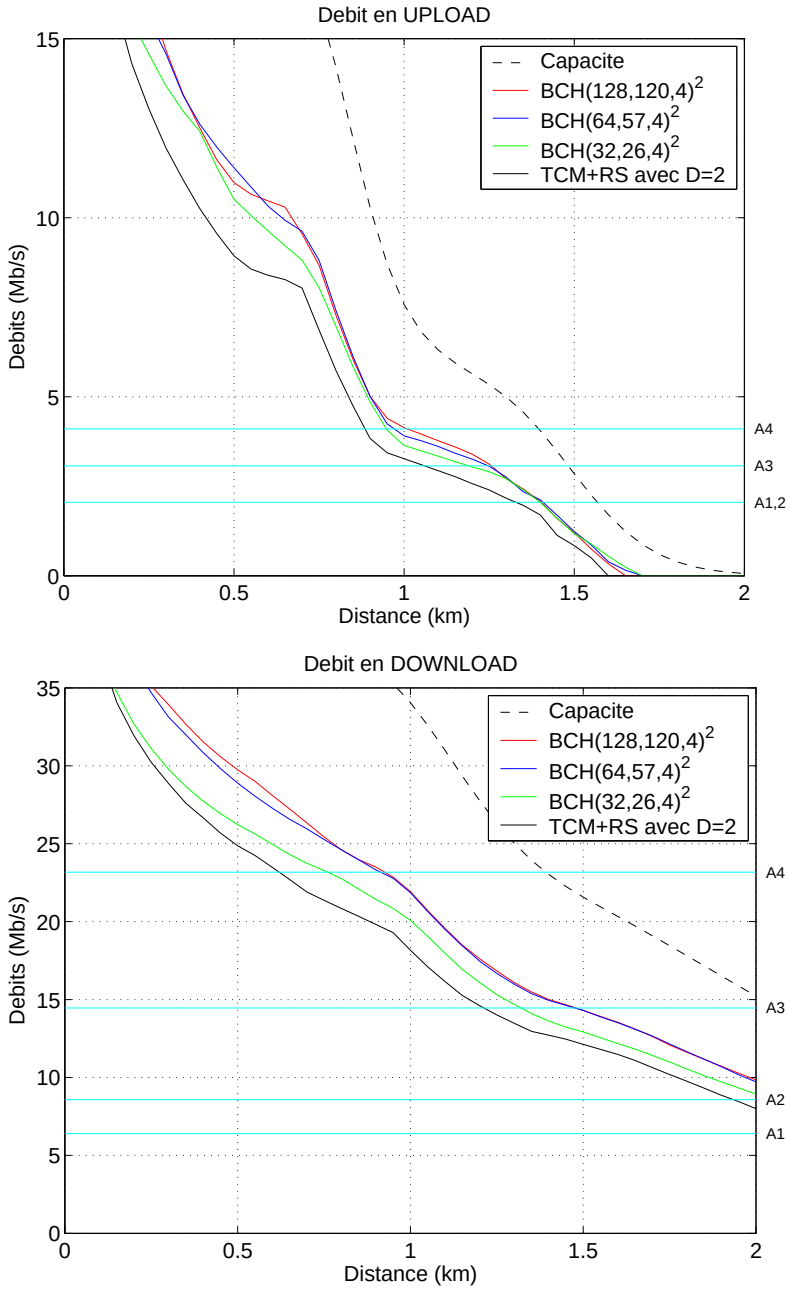


Figure 4.19: Courbe de débit en fonction de la longueur de ligne.

Scénario 5 :

Type de câble :	TP150
Plan de fréquence :	997
Scénario de déploiement :	FTTEx
Nombre de FEXT-NEXT :	49
Alien Noise :	Scénario “E”

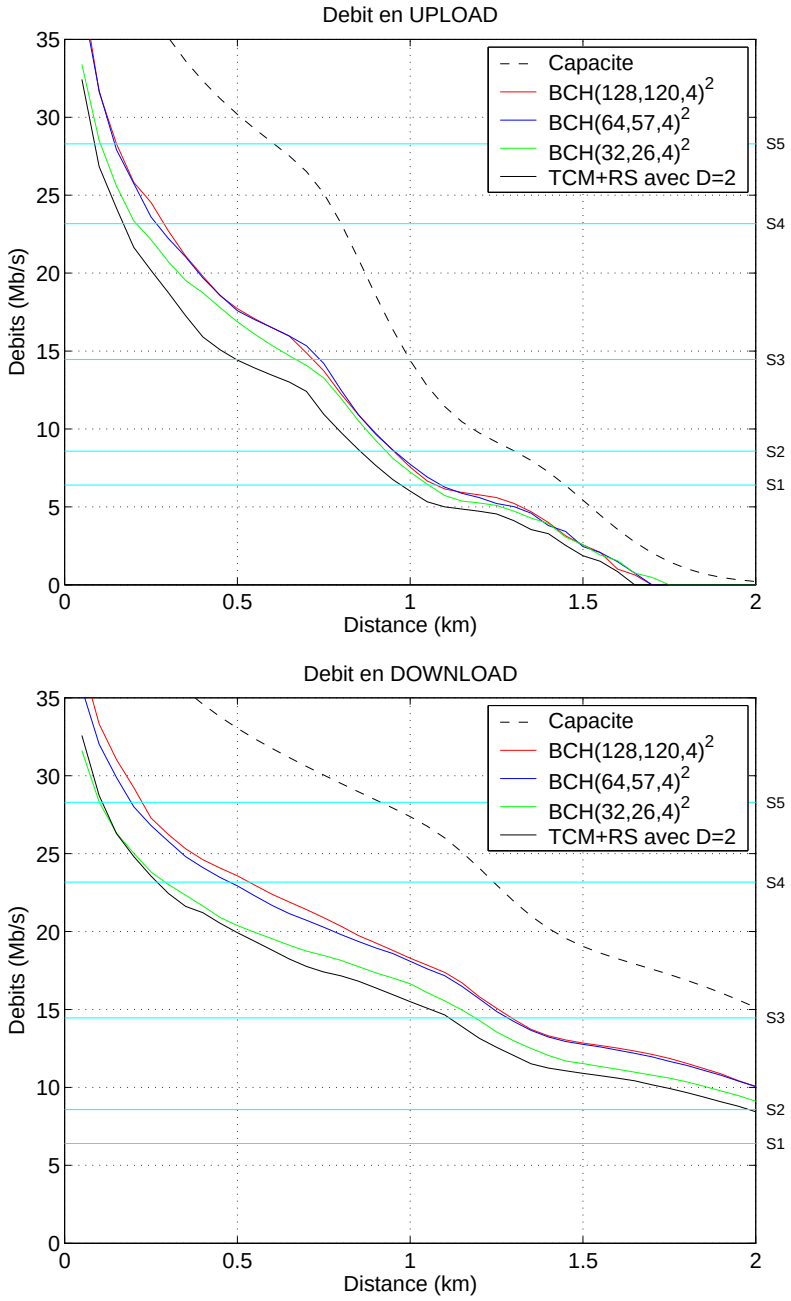


Figure 4.20: Courbe de débit en fonction de la longueur de ligne.

Il apparait clairement sur ces courbes que le code BCH(32,26,4)^2 n'apporte qu'un très faible gain dans la plupart des cas. En effet, tout le gain obtenu au niveau du bitloading est annihilé par son rendement trés faible (voir table 4.3). Par contre, les codes BCH(64,57,4)^2 et BCH(128,120,4)^2 apportent tout deux des gains intéressants, avec un léger avantage pour le code le plus long. Les gains en terme de portée sont plus intéressants pour les profils à haut débit. En particulier, dans le cas du scénario 2, la portée du profil S5 en download passe d'environ 350 m à 700m et passe de 400m à 600m en upload. Dans le cas du scénario 4, la portée du profil A4 passe de 600m à 900m en download. Ce gain en terme de portée signifie tout simplement qu'un plus grand nombre d'utilisateurs pourront profiter d'un profil de débit donné grâce à l'utilisation d'un schéma de codage de type turbo code. Une portée augmentée d'un facteur 2 a pour conséquence un gain d'un facteur 4 en terme de surface couverte et donc d'utilisateurs.

4.5 Conclusion

Ce chapitre avait pour but d'évaluer l'intérêt de l'utilisation éventuelle des turbo codes dans le système VDSL. Nous avons montrés que des gains non négligeables sont apportés par des turbo codes blocs (BCH(128,120,4)^2 et BCH(64,57,4)^2). Le gain apporté permet ainsi d'augmenter la portée des profils de débit définis dans la norme VDSL. Cela permet par conséquent de fournir un débit donné à un plus grand nombre d'utilisateurs et ainsi optimiser l'utilisation des lignes. Le schéma de codage de la première génération du VDSL étant fixé, ce schéma à base de turbo code pourrait être utilisé dans les générations suivantes.

Chapitre 5

Synchronisation de phase

Sommaire

5.1 Introduction

Comme cela a été montré dans le chapitre précédent, le turbo décodage [BG96] procure des performances proches de la capacité de Shannon. Le décodage itératif de codes blocs [HOP96] en particulier permet d'atteindre cette performance remarquable dans le cas de rendements élevés, et Pyndiah [Pyn98] a proposé un algorithme de décodage de turbo codes blocs à faible complexité proche de l'optimum.

Toutefois, ces techniques supposent une synchronisation parfaite par rapport à la phase de la porteuse, ce qui n'est pas réaliste pour les faibles RSB où ces algorithmes de décodage sont efficaces. La moindre erreur de phase a pour conséquence une dégradation importante du TEB (fig. 5.1) et des log-vraisemblances [OC01]. Il faut donc estimer correctement et compenser ces erreurs de phase pour atteindre les performances théoriques.

Ce chapitre a donc pour but de présenter deux algorithmes de synchronisation innovants par le fait qu'ils exploitent les informations souples provenant d'un turbo-décodeur. Le premier [GBBV04c][VBA+03][VGBB04b][VBBG03][VGBB04a] est basé sur la sensibilité des valeurs de fiabilités obtenues par turbo décodage vis-à-vis de la phase de la porteuse et fonctionne en traitant les données par blocs de la taille d'un mot de code. Il estime donc une valeur de phase unique pour le bloc de données. Le deuxième algorithme [GBBV04b] est une boucle adaptative qui utilise les informations souples provenant du turbo décodeur. Il réalise aussi un traitement par bloc de la taille d'un mot de code mais en estimant une phase pour chaque symbole. Il permet donc de poursuivre des évolutions de phase plus rapides. Une première partie définira le modèle utilisé. Dans la deuxième partie sera exposé le principe de l'algorithme de traitement par bloc, les résultats de simulations et les performances. La troisième partie présentera l'algorithme à remodulation souple.

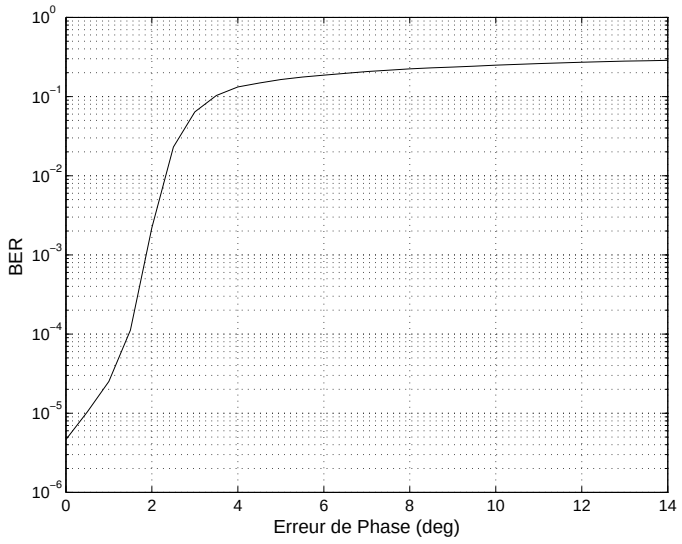


Figure 5.1: Conséquence d'une erreur de phase sur le BER (BCH(32,26,4)^2 sur 1024-QAM avec $E_b/N_0 = 17{,}5$ dB).

5.2 Présentation du modèle

On considère que le canal est idéal et que l'on a uniquement une erreur sur la phase de la porteuse. Supposons que le signal émis soit le suivant :

$$s(t) = Re\left[x\left(t\right)e^{j2\pi f_0 t}\right] = Re\left[s_a\left(t\right)\right] \tag{5.1}$$

avec l'enveloppe complexe :

$$x\left(t\right) = \sum_k a_k h_e\left(t - kT\right) \tag{5.2}$$

avec a_k les symboles émis et $h_e(t)$ l'impulsion de mise en forme à l'émission. L'opération de démodulation est équivalente alors à multiplier le signal analytique $s_a(t)$ par $e^{-j2\pi f_0 t}$ (décalage en fréquence pour obtenir le signal en bande de base). Cependant, il n'est pas possible dans la plupart des systèmes d'obtenir avec exactitude la fréquence f_0 et une phase nulle par rapport à l'émetteur. En effet, l'information sur la phase du signal étant totalement inconnue du récepteur et il est donc nécessaire de l'estimer. Par conséquent, on suppose que la porteuse utilisée pour la démodulation présente un déphasage $\varphi(t)$:

$$y(t) = (s_a(t) + n(t))\, e^{-j(2\pi f_o t - \varphi(t))} \tag{5.3}$$

où $n(t)$ est le bruit présent sur le canal de transmission.
On a donc :

$$y(t) = \sum_k a_k e^{j\varphi(t)} h_e(t - kT) + b(t) \tag{5.4}$$

où $b(t)$ est un bruit supposé gaussien. On suppose que la concaténation du canal et des filtres d'émission et de réception respecte Nyquist et ne crée pas d'intersymbole aux instant d'échantillonnage. Aprés le filtre de réception et l'opération d'échantillonnage (synchronisation rythme idéale), on obtient donc les symboles reçus :

$$y_k = a_k e^{j\varphi_k} + b_k \tag{5.5}$$

où les b_k sont les échantillons d'un bruit gaussien. L'erreur de phase φ_k est supposée avoir une évolution brownienne s'ajoutant à une dérive linéaire [BAG02], c'est-à-dire tel que décrit par l'équation (5.6).

$$\varphi_{k+1} = \varphi_k + \Delta\varphi + w_k \tag{5.6}$$

où $\Delta\varphi$ est une dérive linéaire due à un décalage de fréquence et w_k est un bruit gaussien portant sur la phase du signal (gigue).

Par ailleurs, le système de transmission est encodé par un code produit construit à partir d'un code BCH(31,26,3) étendu par un bit de parité. Le décodeur SISO utilisé pour les itérations du turbo décodage est le décodeur classique de Pyndiah [Pyn98].

5.3 Turbo Synchronisation Par Bloc

5.3.1 Propriété du turbo décodage

On considère, dans un premier temps, que la rotation de phase est constante sur l'ensemble des symboles constituant un mot de code émis, perturbé ensuite par un bruit gaussien. On considère donc le turbo décodage du mot obtenu après le demapping souple des valeurs z_k en sortie du canal données par :

$$z_k = a_k e^{j\varphi} + b_k \tag{5.7}$$

où φ est constant sur un mot de code ($\Delta\varphi$ de l'équation (5.6) est nul). Ce décalage en phase de la porteuse induit une augmentation de la distance euclidienne par rapport au mot émis, et donc une diminution des valeurs de fiabilités (log-vraisemblances) en sortie du turbo décodeur. Il doit donc être possible de mesurer l'effet du déphasage sur les valeurs de fiabilité. Dans [OC01], l'estimation de phase d'une BPSK est basée sur l'information fournie par un turbo décodeur convolutif, moyennée sur 10000 bits. Ici nous allons nous intéresser également au cas des constellations d'ordre supérieur.

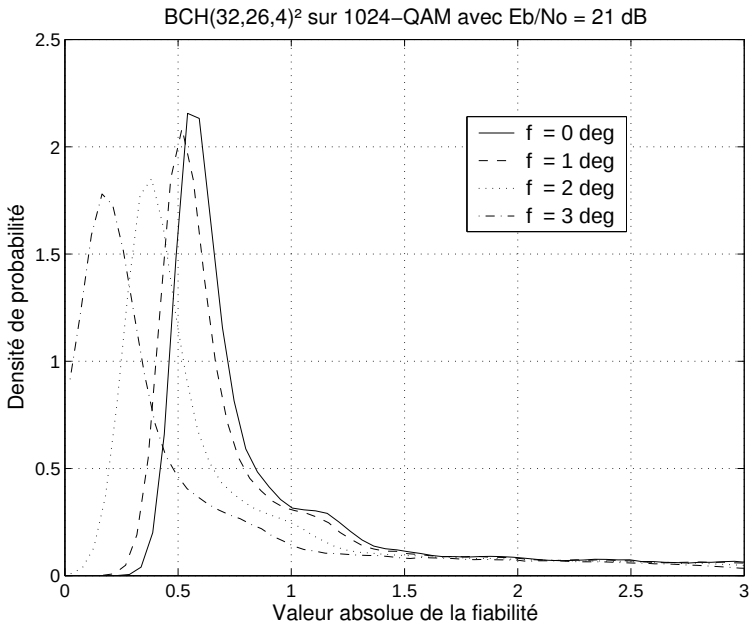


Figure 5.2: Distribution de probabilité des valeurs absolues des fiabilités avec 4 demi-itérations.

Les valeurs de fiabilités les plus faibles sont les plus sensibles à ce phénomène. A contrario, il existe toujours des valeurs de fiabilité de valeur absolue importante, l'ordre de grandeur étant le même que l'on soit bien synchronisé ou pas, car il existera toujours certains bits qui convergeront vers une valeur élevée malgré une mauvaise synchronisation. Une phase correcte assure que peu de valeurs de fiabilités seront faibles si on est dans la zone de convergence du turbo décodeur. Les valeurs de fiabilités les plus faibles tendent vers zéro seulement quand le décalage en phase augmente. Elles apportent donc une information plus pertinente sur le décalage en réception. On peut voir ce phénomène sur la figure 5.2. Il apparait clairement que la distribution pour les valeurs de fiabilités supérieures à 1.5 reste la même quelque soit la valeur du déphasage. En revanche, la distribution pour les valeurs inférieures à 1.5 dépend très fortement de cette erreur de phase.

On définit donc une mesure, notée $M^{(l)}$ qui représente la moyenne des plus petites valeurs de fiabilités à la fin du décodage du l^{eme} mot de code reçu :

$$M^{(l)} = \frac{1}{m_f} \Sigma_{i=1}^{p} Rtri_i^{(l)} \tag{5.8}$$

où m_f est un entier plus petit que la longueur du code et $Rtri^{(l)}$ est le vecteur contenant les valeurs absolues des fiabilités du $l^{ème}$ mot de code triées dans l'ordre croissant. Contrairement à [OC01] qui s'appuie sur les informations extrinsèques, les valeurs de fiabilités $Rtri$ sont les sorties du décodeur SISO triées à partir des valeurs les plus faibles. On utilise ici des valeurs de β dynamiques, comme définies au paragraphe 3.4.6.

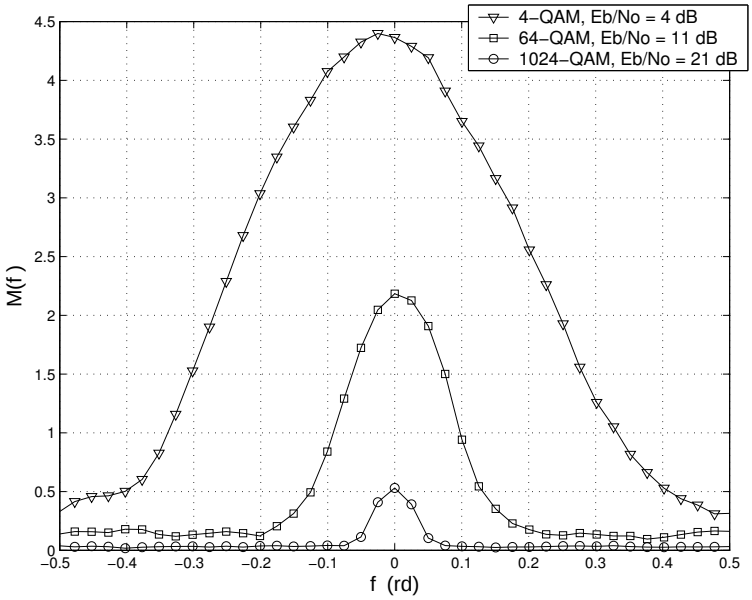


Figure 5.3: Evolution de M pour un mot de code particulier avec un BCH(32,26,4)^2.

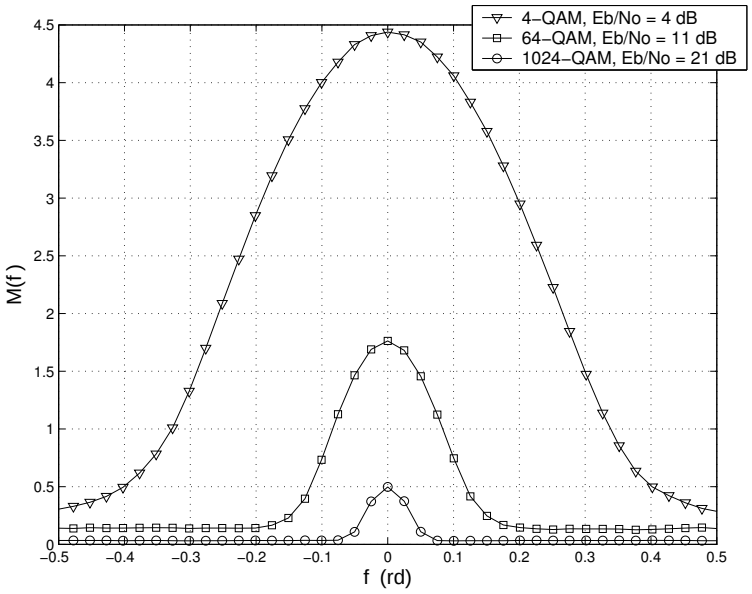


Figure 5.4: Moyenne de M pour 20 mots de code.

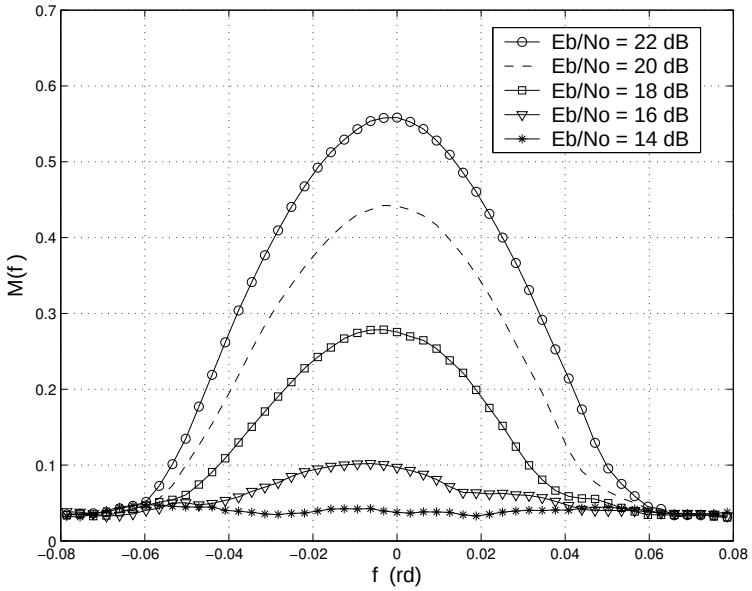


Figure 5.5: Influence de E_b/N_0 sur la forme de $M(\varphi)$sur une 1024-QAM avec un BCH(32,26,4)^2.

Dans les exemples donnés ici, le code utilisé est un BCH(32,26,4)^2. Ce nombre m_f représente un compromis entre un nombre suffisant de termes dans la moyenne et la perturbation apportée par les grandes valeurs de fiabilités. Nous illustrerons comment determiner m_f au paragraphe 5.3.4.4.

Cette mesure $M^{(l)}$ doit donc dépendre de la phase introduite et doit diminuer quand cette phase augmente. Il est nécéssaire d'augmenter le nombre de vecteurs de tests utilisés par le turbo décodage afin de limiter la dépendance de $M^{(l)}$ vis-à-vis du paramètre β qui n'a aucune signification physique. Ceci est très simple à réaliser avec la structure de décodeur simplifié, dans laquelle la quantité de mémoire ne dépend plus du nombre de vecteurs tests. En effet, l'influence de l'approximation utilisée en l'absence de mot concurrent est de perturber le résultat en rendant la courbe discontinue, car cette approximation ne donne pas la valeur réelle de la fiabilité. Le fait d'augmenter le nombre de vecteurs tests a pour effet de diminuer la probabilité de ne pas trouver le mot concurent et donc d'obtenir un critère plus exploitable. Un essai avec l'algorithme de Pyndiah donne la courbe de la figure 5.3 pour un mot de code produit en particulier et donne le résultat de la figure 5.4 si on moyenne sur 20 mots de code.

Malheureusement, on ne peut se contenter d'écarter ces valeurs approximées dans le calcul de la moyenne : en effet, l'approximation a très certainement été utilisée pour certains bits aux premières itérations sans pour autant être utilisée lors la dernière itération. Dans ce cas, le bit en question porte en lui une certaine quantité d'information erronée. Il faut donc diminuer la probabilité d'utiliser cette "information erronée" au cours de chaque itération en augmentant l'espace parcouru par le Chasing. L'algorithme converge rapidement : on se contentera de réaliser 4 demi-itérations par rapport aux 8 demi-itérations utilisées classiquement pour un décodage. Les constantes α que nous

utilisons pour cet algorithme sont les même que pour le décodage canal, et nous avons choisi $\alpha = [0,3; 0,4; 0,45; 0,5; 0,6; 0,7; 0,8]$.

On remarquera également que le comportement de la courbe dépend du SNR (fig.5.5). La présence d'un maximum subsiste à faible SNR tant que l'effet turbo fonctionne, ce qui permettra de s'approcher de la limite de Shannon.

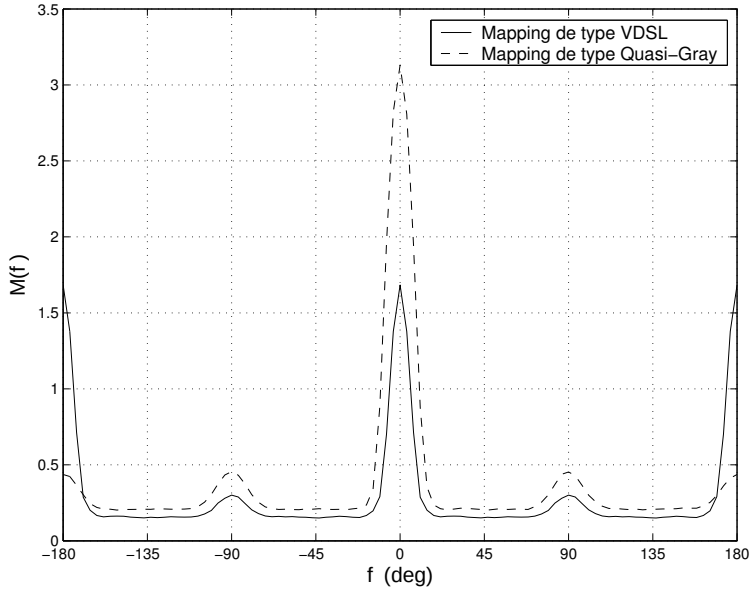


Figure 5.6: Comparaison entre l'étiquetage de type VDSL et l'étiquetage Quasi-Gray sur 64-QAM avec un BCH(32,26,4)^2.

L'étiquetage choisi pour les constellations a également une influence notable sur le résultat obtenu. Si l'on n'y prend pas garde, une ambiguité de π est présente quand l'étiquetage utilisé est de type VDSL tel que défini dans la norme [ANS98]. Ce phénomène est dû à la symétrie présente dans cet étiquetage. En effet, deux points symétriques par rapport au centre de la constellation se trouvent être dans ce cas étiquetés par deux séquences binaires complémentaires. Ainsi, une rotation de π a pour effet d'inverser tous les bits du mot de code. Or, le complément d'un mot de code BCH étendu appartenant également au code, le résultat en sortie du dispositif est une valeur de fiabilité aussi importante que pour le mot de code original reçu sans rotation. On peut voir sur les figures 5.6 et 5.7 l'effet obtenu. Ces deux figures montrent qu'il est possible de s'affranchir de ce problème de deux manières différentes. Une première solution consiste à choisir un étiquetage ne présentant pas ce type de symétrie, et notament l'étiquetage "Quasi-Gray", comme cela apparait sur la figure 5.6. Une autre solution (fig. 5.7), permettant de conserver un étiquetage de type VDSL, consiste à interposer un embrouilleur de la taille d'un mot de code entre le codage canal et l'opération de mapping. Cette deuxième solution a d'autre part pour effet d'atténuer fortement les extrema locaux présents en $-\pi/2$ et en $\pi/2$.

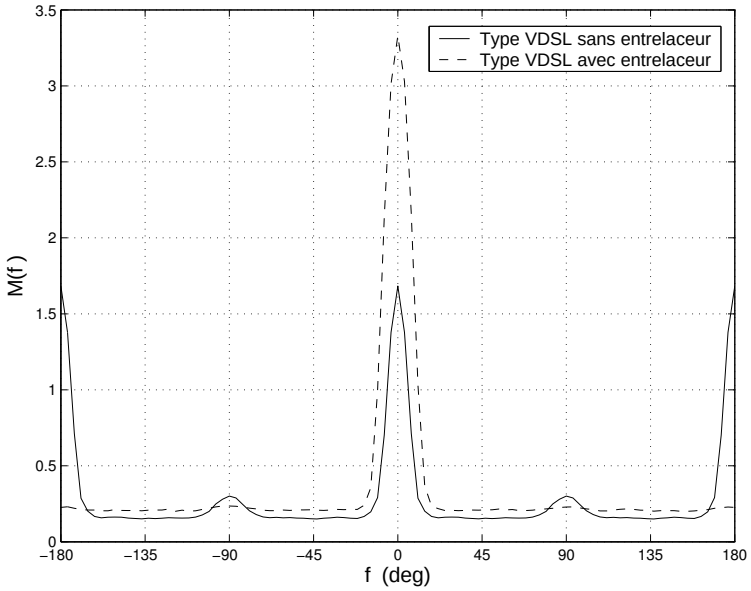


Figure 5.7: Effet d'un embrouilleur dans le système utilisant l'étiquetage VDSL.

D'autre part, on peut toujours observer la présence du pic en présence d'une dérive de phase linéaire. Pour cela, on applique sur les K symboles QAM représentant un mot de code transmis une dérive linéaire (décalage en fréquence Δf), en plus d'une erreur de phase. C'est à dire que pour $k = 1, ..., K$ (K étant le nombre de symboles QAM utilisé pour transmettre un mot de code), on réalise l'opération (5.9).

$$z_k = a_k e^{j\left\{\varphi + \Delta f\left(k - \frac{K}{2}\right)\right\}} + b_k \tag{5.9}$$

Le terme $\frac{K}{2}$ intervient ici pour symétriser la courbe obtenue par rapport au point $\varphi = 0$ et $\Delta f = 0$. On peut donc tracer la courbe $M^{(l)} = f\left(\varphi, \Delta f, a_1, ..., a_K\right)$. Les figures 5.8 et 5.9 montrent donc que le maximum est bel et bien atteint pour $\varphi = 0$ et $\Delta f = 0$ et la largeur des lobes illustre que l'on a donc une aptitude à resister à des dérives de phase.

En d'autres termes, la fiabilité moyennée sur un mot de code est pertinente pour suivre des variations à l'intérieur même d'un mot de code ($\Delta\varphi \neq 0$ dans (5.6)) tant que le modèle des variations reste simple à schématiser. De plus, contrairement à [OC01] , qui moyenne ses fiabilités sur une centaine de mots représentant 10000 bits, la moyenne effectuée ici se fait sur des tailles plus faibles.

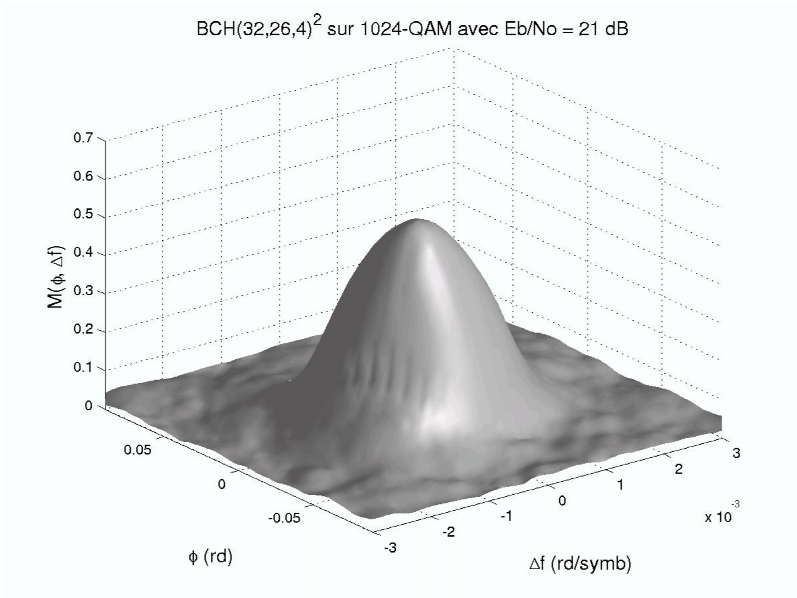


Figure 5.8: Critère $M^{(l)}$ pour un mot de code en particulier.

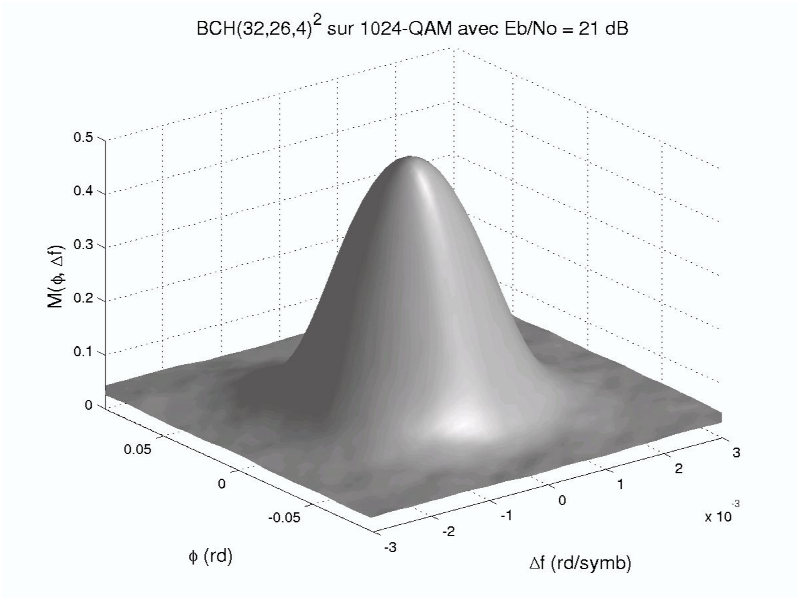


Figure 5.9: Critère $M^{(l)}$ moyenné sur 20 mots de code.

5.3.2 Définition de l'estimateur

Par conséquent, on peut construire un estimateur à partir de cette mesure. En effet, si on suppose que l'on reçoit :

$$y_k = a_k e^{j\varphi} + b_k \tag{5.10}$$

avec φ constant, on peut construire y'_k tel que :

$$y'_k = y_k e^{j\theta} \tag{5.11}$$

et après turbo décodage du $l^{ième}$ mot codé, on obtient $M^{(l)} = f(\theta, y_1, ..., y_K)$ qui dépend d'une variable θ.
Un estimateur est alors défini par :

$$\hat{\varphi} = arg\, max_\theta M^{(l)}(\theta, y_1, ..., y_K) \tag{5.12}$$

Cet estimateur permet donc d'estimer en premier lieu une phase constante sur la longueur d'un mot de code. Sous réserve que le nombre de symboles contenus dans un mot de code ne soit pas trop grand, ce critère permettra également de poursuivre une évolution de phase telle que décrite par l'équation (5.6). Il suffit pour cela que la dérive soit telle que l'on reste dans le pic central illustré par les figures 5.8 et 5.9.

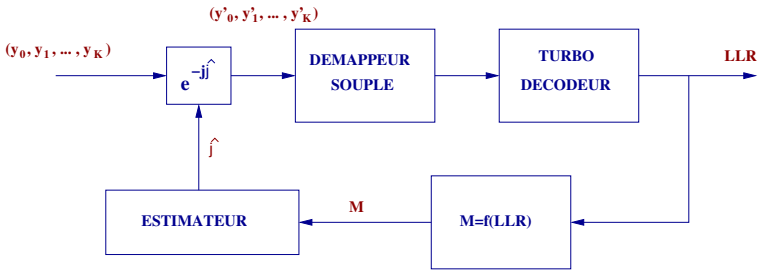


Figure 5.10: Schéma de principe de la turbo synchronisation.

5.3.3 Description de l'algorithme

Si l'on suppose qu'à l'initialisation, le recepteur n'a pas de connaissance a priori sur la valeur de la phase, il est possible de calculer simultanément plusieurs $M^{init}(\theta_i)$ correspondant à plusieurs valeurs d'essais de θ_i équiréparties. La valeur de la phase θ_i correspondant à la plus grande valeur de $M^{init}(\theta_i)$ peut ensuite être choisie comme initialisation de l'algorithme.

5.3.3.1 Gradient

L'estimation de la phase revient donc à rechercher le maximum de $M^{(l)}(\theta, Y)$ avec $Y = (y_1, ..., y_K)$. Pour cela, nous utilisons le système représenté sur la figure 5.10.

On suppose tout d'abord que la phase est constante sur la longueur du mot de code, ce qui est raisonnable pour un mot de code relativement court. Le turbo décodeur est inclus dans la partie droite du schéma. Les échantillons du signal QAM bruités et démodulés Y sont multipliés par le terme $e^{-j\hat{\varphi}}$ de façon à corriger un éventuel défaut de phase, où $\hat{\varphi}$ est la phase estimée. Sous la condition d'être dans le cône de convergence de

l'algorithme, la valeur maximale de $M^{(l)}(\theta, Y)$ sur le $l^{ème}$ mot de code peut être atteinte avec un algorithme du gradient stochastique.

Dans ce cas, l'algorithme de recherche de la phase de la porteuse est :

$$\hat{\varphi}_i = \hat{\varphi}_{i-1} + \mu \left(M^{(l)}\left(\hat{\varphi}_{i-1}, y_1, ..., y_K\right) - M^{(l)}\left(\hat{\varphi}_{i-2}, y_1, ..., y_K\right) \right) \left[sign\left(\hat{\varphi}_{i-2} - \hat{\varphi}_{i-1}\right) \right] \quad (5.13)$$

où classiquement la valeur du pas μ est choisie de façon à réaliser un compromis entre l'erreur quadratique moyenne et la vitesse de convergence. La valeur choisie pour initialiser l'algorithme à chaque nouveau mot de code est la valeur qui a été estimée pour le mot de code précédent.

Dans la pratique, il y a souvent un décalage en fréquence Δf non négligeable entre la porteuse de l'emetteur et celle du récepteur ; il est alors possible de prendre en compte ce décalage dans l'estimation. En effet, une bonne estimation de la dérive peut être obtenue sur la base des dernières estimations de phase.

5.3.3.2 Identification avec une parabole

L'algorithme précédent peut cependant demander un nombre d'estimations de $M^{(l)}(\theta, Y)$ relativement important, ce qui implique autant de rephasages et de demapping avant les décodages. Une variante de l'algorithme plus légère en temps de calcul que le gradient peut être obtenue en modélisant l'évolution de $M^{(l)}(\theta, Y)$ par une simple parabole [eLP94] dans son cône de convergence (cf figure 5.4). Dans ce cas, seules trois évaluations sont suffisantes pour obtenir $\hat{\varphi}$. En considérant φ_0 et une constante $\Delta\varphi$, et en définissant $\varphi_1 = \varphi_0 - \Delta\varphi$ et $\varphi_2 = \varphi_0 + \Delta\varphi$, le maximum de la parabole passant par ces trois points est donné par :

$$\hat{\varphi} = \varphi_0 + \Delta\varphi \frac{M^{(l)}(\varphi_2) - M^{(l)}(\varphi_1)}{4M^{(l)}(\varphi_0) - 2M^{(l)}(\varphi_2) - 2M^{(l)}(\varphi_1)} \quad (5.14)$$

φ_0 est la valeur de l'estimation obtenue pour le mot de code précédent. Cette méthode permet une économie importante en terme de calcul, et permet une parallélisation de celui-ci.

Cependant, la courbe parcourue par le critère que nous utilisons n'étant pas une vraie parabole, ce résultat n'est qu'une approximation et n'est pas indépendant de $\Delta\varphi$. Ainsi, il existe une valeur optimale de ce paramètre pour lequel l'estimation sera la meilleure possible. Dans notre cas, une EQM optimale est obtenue pour $\Delta\varphi$ égal à 20% de la largeur du lobe.

Cependant, dans certaines circonstances, notamment dans le cas où les perturbations sur la phase sont importantes, cet algorithme présente des "décrochements". En effet, la forme du critère ne peut être approximée par une parabole uniquement dans la zone autour du maximum. Si les trois mesures sont faites sur un "bord" de la courbe, il est probable que le résultat soit aberrant et conduise à une divergence de l'algorithme.

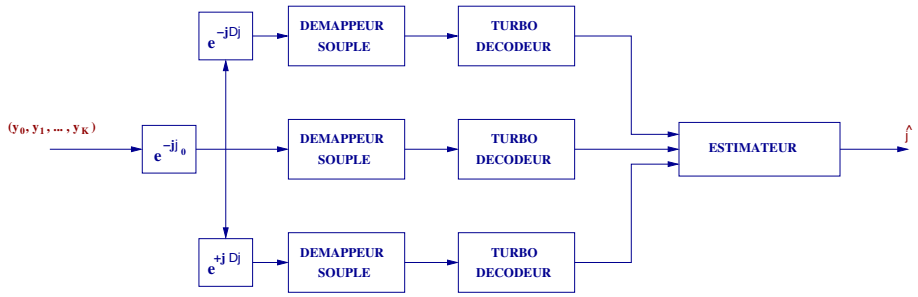


Figure 5.11: Algorithme rapide

5.3.3.3 Identification avec une gaussienne

On cherche ici à améliorer la modélisation de l'évolution de $M^{(l)}$ en fonction du déphasage. On identifie cette évolution avec une gaussienne [eLP94] de valeur maximale M_{max}, de valeur minimale M_{min}, centrée sur θ_0, et dont la largeur est paramétrée par K :

$$M_G(\theta) = (M_{max} - M_{min})\, e^{-K(\theta-\theta_0)^2} + M_{min} \tag{5.15}$$

La figure 5.12 représente cette modélisation. Le but est alors de choisir les paramètres de la gaussienne de manière à minimiser l'erreur quadratique entre les mesures et le modèle. En considérant k valeurs de phase $\varphi_1, \ldots, \varphi_k$, et k mesures $M^{(l)}(\varphi_1), \ldots, M^{(l)}(\varphi_k)$ on définit donc l'erreur quadratique :

$$\varepsilon = \sum_{j=1}^{j=k} \left(M_G(\varphi_j) - M^{(l)}(\varphi_j)\right)^2 \tag{5.16}$$

Il est alors possible de trouver les valeurs des paramètres qui minimisent cette erreur quadratique par un simple algorithme du gradient :

$$\begin{aligned}
M_{max}^{i+1} &= M_{max}^{i} - \lambda_i \left.\frac{\partial \varepsilon}{\partial M_{max}}\right|_{M_{max}^i, M_{min}^i, \theta_0^i, K^i} \\
M_{min}^{i+1} &= M_{min}^{i} - \lambda_i \left.\frac{\partial \varepsilon}{\partial M_{min}}\right|_{M_{max}^i, M_{min}^i, \theta_0^i, K^i} \\
\theta_0^{i+1} &= \theta_0^{i} - \lambda_i \left.\frac{\partial \varepsilon}{\partial \theta_0}\right|_{M_{max}^i, M_{min}^i, \theta_0^i, K^i} \\
K^{i+1} &= K^{i} - \lambda_i \left.\frac{\partial \varepsilon}{\partial K}\right|_{M_{max}^i, M_{min}^i, \theta_0^i, K^i}
\end{aligned} \tag{5.17}$$

où interviennent les dérivées analytiques déduites de (5.15) et (5.16). Cet algorithme ne demande pas une quantité de calcul importante après le calcul des $M^{(l)}(\varphi_j)$. Il converge en quelques itérations et on a alors directement une estimation de la phase par :

$$\hat{\varphi} = \theta_0 \tag{5.18}$$

On peut se contenter de 4 mesures pour que cet algorithme soit efficace. Il a l'avantage d'être paramétrable au niveau du nombre de mesures et de leur distribution. Il est ainsi possible d'obtenir un algorithme trés robuste aux variations de phase simplement en augmentant le nombre de tests $M^{(l)}(\varphi_j)$. C'est cette modélisation que nous utiliserons.

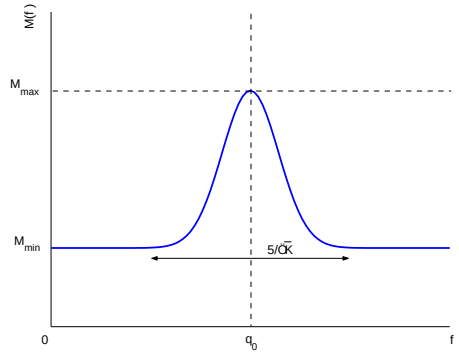


Figure 5.12: Modélisation par une gaussienne.

5.3.4 Etude des performances

5.3.4.1 Dérive linéaire

La figure 5.13 illustre la capacité de poursuite dans le domaine temporel de cette turbo synchronisation dans le cas d'une dérive $\Delta\varphi = 2\pi.10^{-4}$ radian par symbole 1024 QAM et une gigue de phase nulle dans l'équation (5.6). Le RSB est fixé à 21 dB. La phase estimée est proche de la dérive comme cela était prévisible, vu la largeur du lobe de la figure 5.8. Dans ce cas, on estime la dérive moyenne de phase en tenant compte des estimations de phase précédentes. Une simple moyenne sur la différence de phase entre plusieurs mots de code précédents donne une bonne estimation de cette dérive.

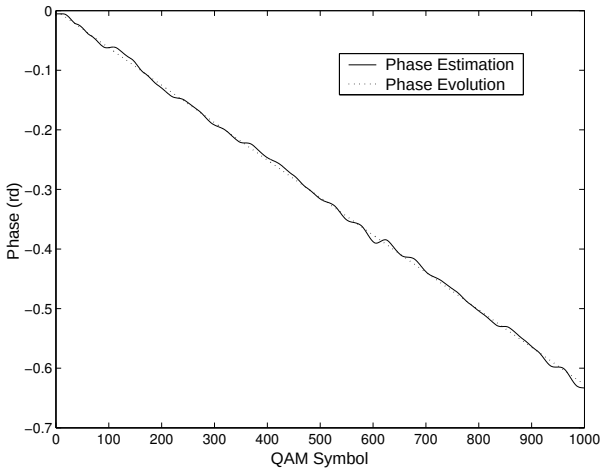


Figure 5.13: Poursuite d'une dérive de phase sur une 1024-QAM avec un BCH(32,26,4)^2.

5.3.4.2 Erreur quadratique moyenne

Nous allons comparer dans la suite notre algorithme de turbo synchronisation avec un algorithme classique appelé puissance 4 [MdJ94]. Cet algorithme aveugle est fréquemment utilisé dans le cas de constellations QAM. Si on note l'observation par $y_k = a_k e^{j\varphi_k} + b_k$, alors l'estimateur puissance 4 est basé sur l'égalité $\varphi = \frac{1}{4} arg\left(E\left(y_k^4\right)\right) + \frac{\pi}{4}$ et sa mise en oeuvre adaptative [BAG02] se fait par :

$$\hat{\varphi}_k = \hat{\varphi}_{k-1} + \gamma \cdot Im\left(y_k^4 e^{-j4\hat{\varphi}_{k-1}}\right) \tag{5.19}$$

Elle converge vers la solution (avec une ambiguité de $\pi/2$) quand on estime une phase constante.

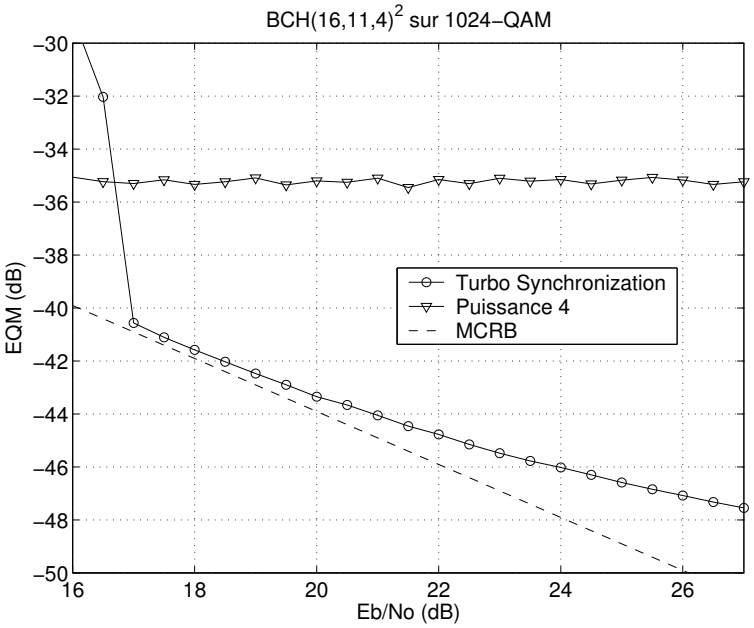


Figure 5.14: Erreur quadratique moyenne sur une 16-QAM.

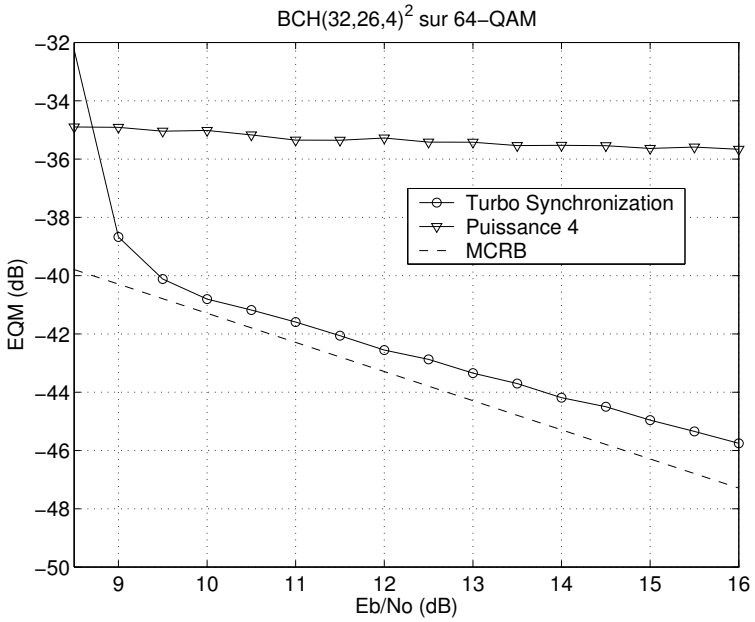


Figure 5.15: Erreur quadratique moyenne sur une 64-QAM.

La borne de Cramer-Rao (CRB) est un moyen pour évaluer les performances d'un estimateur de façon absolue. Cette borne définit les performances optimales que l'on peut obtenir sur un problème d'estimation donné. La difficulté du calcul de la CRB relative aux paramètres de synchronisation est liée à la présence des symboles a_k inconnus assimilables à un bruit multiplicatif.

En supposant l'équiprobabilité des symboles de la constellation, la CRB exacte peut être numériquement obtenue en utilisant une intégration par rapport aux points de la constellation lors du calcul de la vraisemblance des observations [NSM03]. La symétrie des constellations QAM carrées permet d'exprimer la vraisemblance d'une observation y_k comme :

$$f\left(y_k,\varphi\right)=e^{\frac{-|y_k|^2}{2\sigma^2}}\sum_{Q_i\in C_1}e^{\frac{-|Q_i|^2}{2\sigma^2}}\left[2cosh\left(\frac{Re\left(y_kQ_i^*e^{-j\varphi}\right)}{\sigma^2}\right)+2cosh\left(\frac{Re\left(y_kQ_ie^{-j\varphi}\right)}{\sigma^2}\right)\right] \tag{5.20}$$

où l'ensemble C_1 contient les symboles Q_i du premier quadrant, et σ^2 désigne la variance du bruit. Le calcul de la CRB devient rapidement laborieux pour les constellations de plus en plus grandes. Pour contourner ce problème, une borne modifiée MCRB [DMR94][GRM98][Moe98] utilise un report de la marginalisation par rapport aux symboles lors du calcul de la matrice de Fisher par :

$$MCRB\left(\varphi\right)^{-1}=-E_{y,a}\left(\frac{\partial^2log\left(f\left(y/a,\varphi\right)\right)}{\partial\varphi^2}\right) \tag{5.21}$$

avec a l'ensemble des séquences de symboles QAM possibles sur les N_s symboles observés. Ceci permet d'obtenir des expressions plus faciles à mettre en oeuvre. Dans le cas d'une erreur de phase et en supposant les symboles équiprobables, $MCRB(\varphi) = \frac{\sigma^2}{N_s}$ où N_s désigne le nombre de symboles observés et σ^2 la variance du bruit d'observation.

Les figures 5.14 à 5.16 comparent l'algorithme de turbo synchronisation avec la $MCRB$ et l'algorithme puissance 4 en terme d'erreur quadratique moyenne (EQM) avec une gigue de phase w (voir équation 5.6) d'écart-type $2\pi.10^{-4}$rd et une dérive nulle.

On voit sur ces figures que l'algorithme se rapproche de la borne pour une valeur de SNR donnée, et qu'ensuite il s'en éloigne. En effet, cet algorithme étant basé sur les valeurs de fiabilités fournies par le turbo décodeur, il ne fonctionne que si celui ci est dans sa zone de convergence, ce qui explique la divergence pour les faibles SNR. D'autre part, l'algorithme utilisé dans dans cette simulation a été optimisé (nombre et distribution des mesures du paragraphe 5.3.3.3) pour un point de fonctionnement donné, qui correspond à $E_b/N_0 = 10dB$ pour la 64-QAM et $E_b/N_0 = 19dB$ pour la 1024-QAM. Or, les performances de l'algorithme dépendent du nombre de mesures utilisées pour l'identification des paramètres. Pour atteindre les performances prédites par la borne (-60 dB pour un SNR de 28 dB sur la figure 5.16), il faut une très grande précision. Cela nécessiterait un nombre de mesures beaucoup plus important pour modéliser plus finement la courbe $M^{(l)}(\varphi)$. On s'éloigne donc petit à petit de l'optimum en s'éloignant du point de fonctionnement.

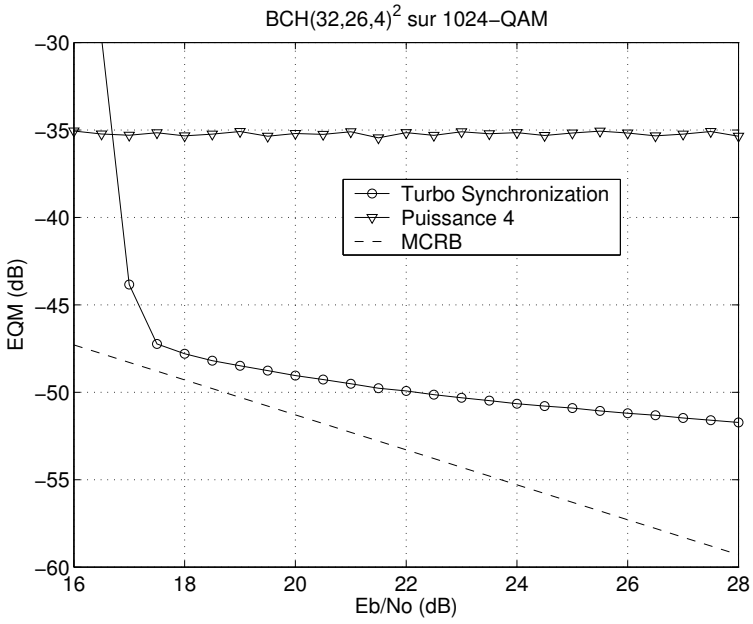


Figure 5.16: Erreur quadratique moyenne sur une 1024-QAM.

5.3.4.3 Taux d'erreur binaire

Les figures 5.17 à 5.18 comparent les Taux d'Erreur Binaires obtenus dans le cas idéal, avec la turbo synchronisation et avec l'algorithme puissance 4 (5.19).

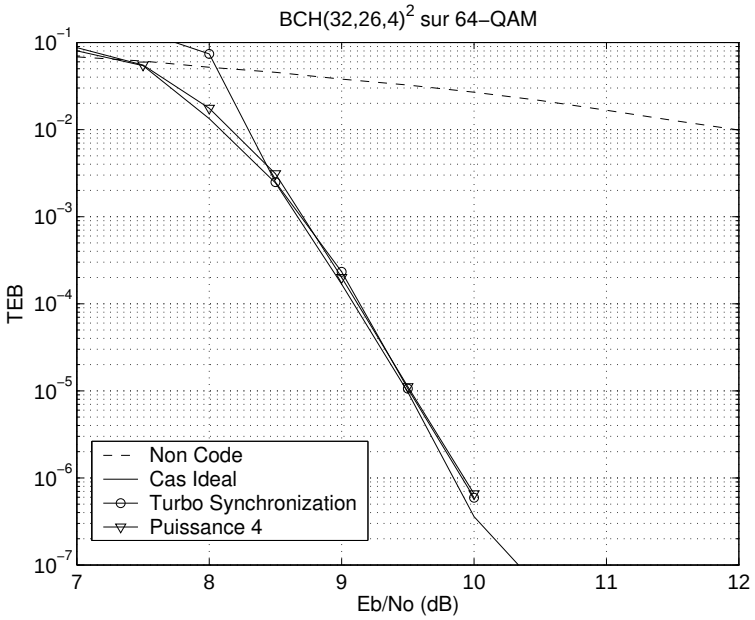


Figure 5.17: Taux d'erreur binaire pour une 64-QAM.

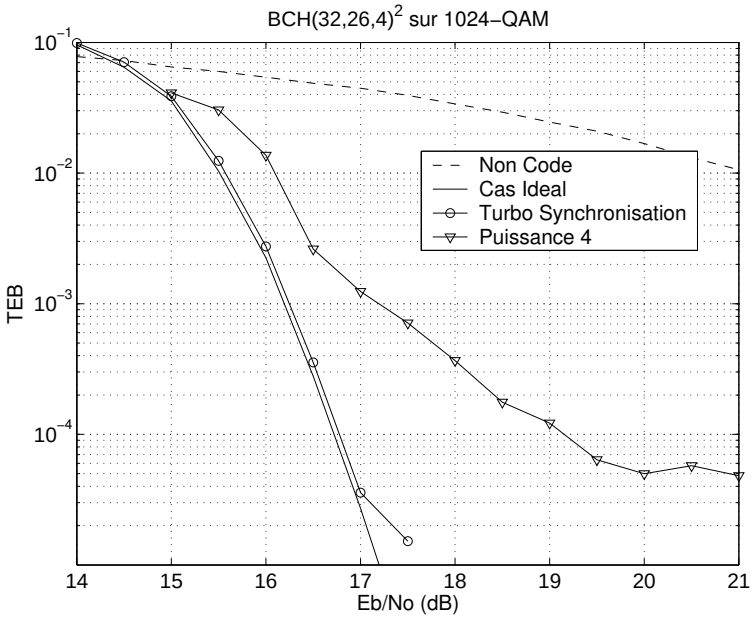


Figure 5.18: Taux d'erreur binaire pour une 1024-QAM.

L'algorithme puissance 4 n'est manifestement pas suffisant dans les conditions de simulation de la figure 5.18, c'est-à-dire pour une constellation de grande taille et une perturbation brownienne sur la phase d'écart-type $2\pi 10^{-4}$ radian. Ceci est dû au bruit

autogénéré par cet algorithme aveugle [BAG02]. Par contre, l'algorithme de Turbo Synchronisation permet de se rapprocher du cas de la synchronisation idéale, c'est-à-dire le cas où il n'y a pas perturbation de phase.

5.3.4.4 Influence du paramètre m_f

Nous cherchons ici à étudier l'influence du nombre de bits utilisés dans la moyenne sur l'erreur quadratique moyenne obtenue par l'estimateur (cf equation (5.8)). Pour cela, des simulations ont été réalisées sur un large panel de valeurs et pour deux longueurs de code différentes. Dans un premier temps, on peut voir sur la figure 5.19 l'influence du paramètre m_f sur la mesure $M^{(l)}(\varphi)$, moyenné sur 20 mots de code. On remarque que si m_f atteint une valeur trop importante, le rapport entre la valeur maximum et la valeur minimum du critère tend à diminuer. La dynamique de cette mesure devient donc plus faible. On observe également des perturbations sur la courbe, qui n'est plus aussi lisse que dans les cas où m_f est faible. Ce phénomène est dû aux grandes valeurs de fiabilité, peu sensibles au déphasage φ.

La courbe 5.20 montre le comportement de l'erreur quadratique moyenne en fonction du rapport signal à bruit selon la quantité m_f de bits utilisés dans le calcul de la mesure $M^{(l)} = f(\theta, y_1, ..., y_n)$. La valeur idéale de m_f représente un compromis entre la quantité d'information utilisée et les perturbations apportées par les grandes valeurs.

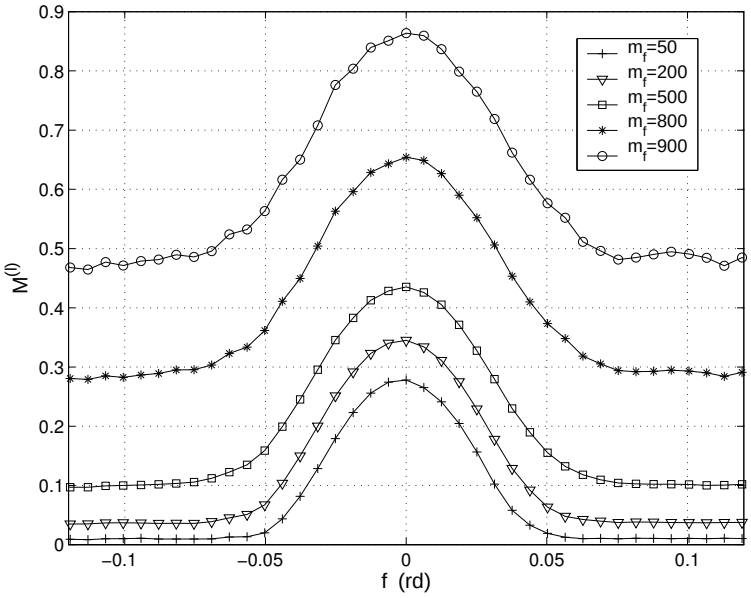


Figure 5.19: Influence de m_f sur la mesure $M(\varphi)$, pour un code BCH(32,26,4)^2 sur 1024-QAM à 19 dB.

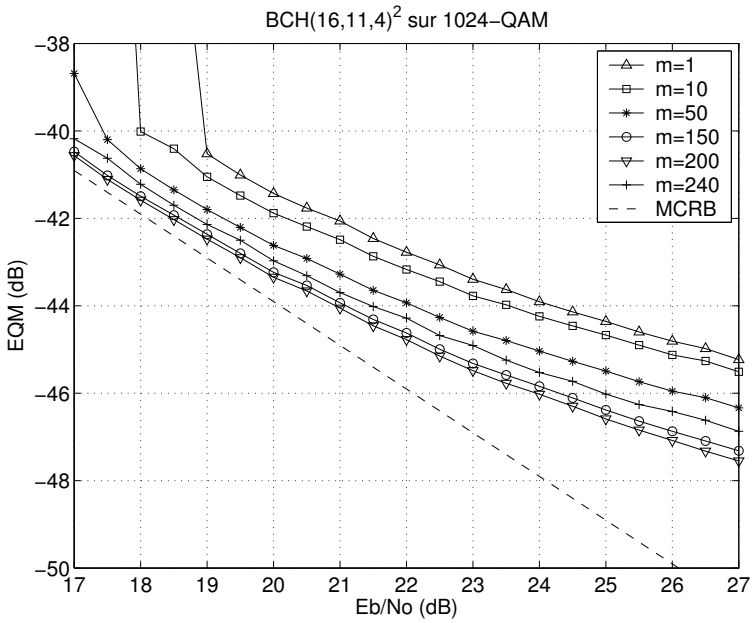


Figure 5.20: EQM en fonction du E_b/N_0 obtenue pour plusieurs valeurs de m_f avec un code BCH(16,11,4)^2 sur 1024-QAM.

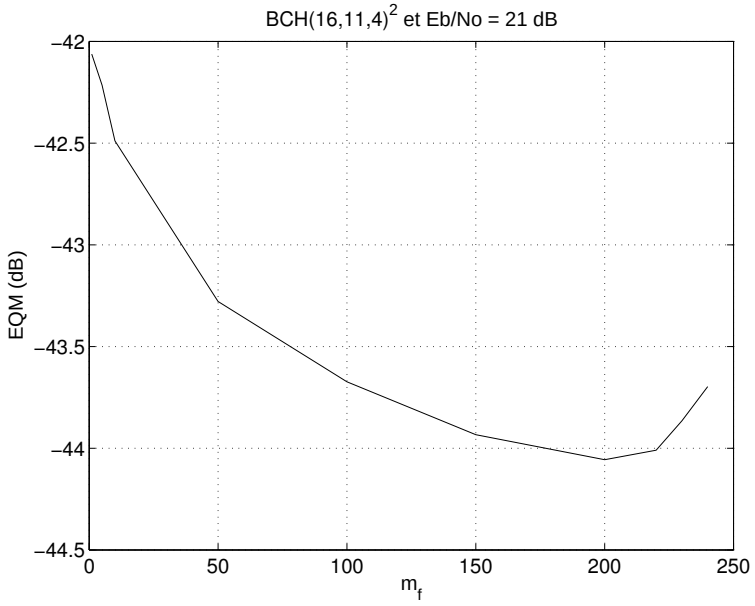


Figure 5.21: EQM en fonction de m_f pour $E_b/N_0 = 21$ dB (BCH(16,11,4)^2 sur 1024-QAM).

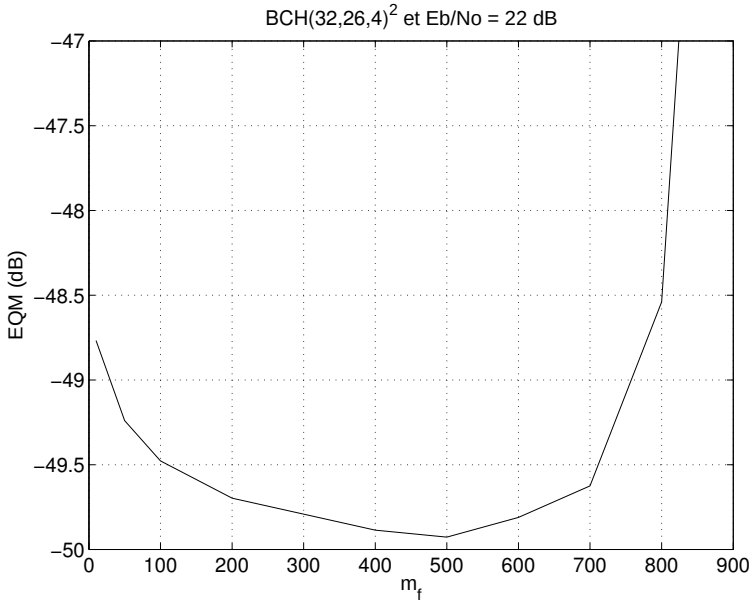


Figure 5.22: EQM en fonction de m_f pour $E_b/N_0 = 22$ dB (BCH(32,26,4)^2 sur 1024-QAM).

Les figures 5.21 et 5.22 représentent l'erreur quadratique moyenne obtenue pour un rapport signal à bruit fixé en fonction du paramètre m_f. Il apparait que pour le code BCH(16,11,4)^2 on peut utiliser environ 80 % des bits pour calculer la moyenne des fiabilités, alors que dans le cas du code BCH(32,26,4)^2 l'optimum se situe plutôt autour des 50 %. Cela est dû au fait que le second code est plus long et donc présente lors de son décodage une proportion de mots concurrents manquants plus importante. De ce fait les valeurs approximées dans ces positions tendent à perturber la moyenne. Par conséquent, plus un code sera long, plus la proportion de bits utilisables dans la moyenne sera faible.

5.4 Boucle à remodulation souple

Le but de cette partie est de proposer une classe d'algorithmes d'estimation adaptative des paramètres d'une communication utilisant une remodulation souple. En effet, comme nous l'avons vu avec l'exemple des turbo codes, la théorie de l'information permet d'affirmer que le fait d'exploiter l'information souple [Sha48][Dup89] provenant du canal améliore les performances par rapport au cas où des décisions dures sont utilisées. Les informations a priori peuvent être utilisées dans les boucles présentées dans ce chapitre pour améliorer la qualité de l'estimation du ou des paramètres d'intérêt.

Dans [LM04], un algorithme d'estimation de phase exploitant les informations souples provenant d'un turbo décodeur est présenté. Cet algorithme estime une phase constante sur la longueur d'un mot de code en exploitant les log-vraisemblances des symboles.

Dans notre cas, la phase n'est pas supposée constante sur un mot. Une boucle adaptative permet de remettre à jour l'estimation de la phase pour chacun des symboles

sur lesquels le mot de code est mappé. On suppose ici que le système utilisé dispose d'un turbo décodeur fournissant des log-vraisemblances pour chacun des bits reçus. Etant donné que les informations sont dans ce cas traitées par bloc, il est intéressant de tirer parti de cette information souple pour affiner l'estimation des paramètres du canal (phase et gain), et ainsi améliorer les performances globales du système. Ces algorithmes de poursuite sont d'ailleurs construits dans le but de s'appliquer à n'importe quelle forme de constellation, et par exemple les constellations carrées de type QAM.

Dans un premier temps, j'exposerai les developpements mathématiques dans le cas d'une simple boucle de phase puis d'une boucle de gain. Ensuite je discuterai du problème de l'estimation conjointe de ces deux paramètres. Enfin, je présenterai les résultats de simulation.

5.4.1 Boucle de phase

5.4.1.1 Modèle

On considère que l'on observe un bloc de données complexes à la sortie d'un canal. Les symboles émis ont subi une rotation du fait de l'erreur de phase et sont perturbés par un bruit blanc gaussien additif :

$$y_k = a_k e^{i\theta_k} + b_k \tag{5.22}$$

pour $k = 1 \cdots K$ avec $a_k \in \{Q_1, Q_2, \cdots, Q_M\}$ qui sont les symboles indépendants d'une constellation M-QAM et $M = 2^N$.

En groupant les observations dans un vecteur $Y = (y_1, \cdots, y_K)^T$, il est possible d'exprimer le vraisemblance des observations grâce à l'hypothèse du bruit gaussien, c'est à dire :

$$P\left(Y \mid \theta, a_1, \cdots, a_K\right) = \prod_{k=1}^{K} \frac{1}{\pi\sigma_b^2} \exp\left(-\frac{\left|y_k - a_k e^{i\theta}\right|^2}{\sigma_b^2}\right) \tag{5.23}$$

5.4.1.2 Vraisemblance des observations avec information a priori

On suppose que l'on a une information a priori sur les symboles a_k inconnus. Il est donc possible d'intégrer sur ces symboles inconnus (en considérant les a_k indépendants) :

$$\begin{aligned} P(Y \mid \theta) &= \sum_{\substack{a_k \in \{Q_1, \cdots, Q_M\} \\ k = 1 \cdots K}} P\left(Y \mid \theta, a_1, \cdots, a_K\right) \times P\left(a_1, \cdots, a_K\right) \\ &= \sum_{\substack{a_k \in \{Q_1, \cdots, Q_M\} \\ k = 1 \cdots K}} \prod_{k=1}^{K} P\left(a_k\right) \frac{1}{\pi\sigma_b^2} \exp\left(-\frac{\left|y_k - a_k e^{i\theta}\right|^2}{\sigma_b^2}\right) \end{aligned} \tag{5.24}$$

d'où, aprés factorisation :

$$P(Y \mid \theta) = \left(\frac{1}{\pi\sigma_b^2}\right)^K \prod_{k=1}^{K} \left[\sum_{m=1}^{M} P\left(a_k = Q_m\right) \exp\left(-\frac{\left|y_k - Q_m e^{i\theta}\right|^2}{\sigma_b^2}\right)\right] \tag{5.25}$$

Or en considérant que les bits sont indépendants, on peut exprimer la probabilité d'un symbole en fonction de la probabilité de chaque bit :

$$P\left(a_k = Q_m\right) = \prod_{n=1}^{N} P\left(b_n^k = q_n^m\right) \tag{5.26}$$

où b_n^k est le $n^{ème}$ bit du $k^{ème}$ symbole QAM a_k et q_n^m est le $n^{ème}$ bit du symbole QAM Q_m appartenant à la constellation considérée et les bits étant dans $\{+1, -1\}$. On peut alors exprimer la probabilité d'un bit en fonction de sa log-vraisemblance tel que le montre les équations suivantes :

$$P(b_n^k = +1) = \frac{e^{\frac{L_n^k}{2}}}{e^{-\frac{L_n^k}{2}} + e^{\frac{L_n^k}{2}}} \text{ et } P(b_n^k = -1) = \frac{e^{-\frac{L_n^k}{2}}}{e^{-\frac{L_n^k}{2}} + e^{\frac{L_n^k}{2}}} \tag{5.27}$$

où L_n^k est la log-vraisemblance sur le n^{eme} bit du k^{eme} symbole QAM émis. On définit $L^k = \left(L_1^k, ..., L_N^k\right)$.

Il en découle l'expression suivante :

$$P(b_n^k = q_n^m) = \frac{e^{\frac{q_n^m L_n^k}{2}}}{2\cosh\left(\frac{L_n^k}{2}\right)} \tag{5.28}$$

Ainsi, en utilisant la relation précédente, l'équation (5.26) peut se mettre sous la forme :

$$P\left(a_k = Q_m\right) = \left[\prod_{n=1}^{N} \frac{1}{2\cosh\left(\frac{L_n^k}{2}\right)}\right] \times \exp\left(\frac{1}{2}\sum_{n=1}^{N} q_n^m L_n^k\right) \tag{5.29}$$

Par conséquent, l'expression (5.25) devient :

$$P\left(Y \mid \theta\right) = \left(\frac{1}{\pi\sigma_b^2}\right)^K \prod_{k=1}^{K} \left[\left(\prod_{n=1}^{N} \frac{1}{2\cosh\left(\frac{L_n^k}{2}\right)}\right) \exp\left(-\frac{|y_k|^2}{\sigma_b^2}\right) \sum_{m=1}^{M} \exp\left(\frac{1}{2}\sum_{n=1}^{N} q_n^m L_n^k + \frac{2Re\left(y_k \overline{Q_m} e^{-i\theta}\right) - |Q_m|^2}{\sigma_b^2}\right)\right] \tag{5.30}$$

Afin de simplifier les notations, on posera :

$$W_m\left(y_k, L^k, \theta\right) = exp\left(\frac{1}{2}\sum_{n=1}^{N} q_n^m L_n^k - \frac{\left|y_k - e^{i\theta} Q_m\right|^2}{\sigma_b^2}\right) \tag{5.31}$$

Nous donnerons une interprétation de cette grandeur au paragraphe 5.4.1.4.

L'équation (5.30) se met alors sous la forme :

$$P\left(Y \mid \theta\right) = \left(\frac{1}{\pi\sigma_b^2}\right)^K \prod_{k=1}^{K} \left[\left(\prod_{n=1}^{N} \frac{1}{2\cosh\left(\frac{L_n^k}{2}\right)} \right) \sum_{m=1}^{M} W_m\left(y_k, L^k, \theta\right) \right] \tag{5.32}$$

5.4.1.3 Calcul de la dérivée

Dans le but de réaliser un estimateur de la phase, on calcule la dérivée du logarithme de la vraisemblance obtenue en (5.32) :

$$\begin{aligned} \frac{\partial}{\partial\theta} \log P\left(Y \mid \theta\right) &= \sum_{k=1}^{K} \frac{\partial}{\partial\theta} \log \sum_{m=1}^{M} W_m\left(y_k, L^k, \theta\right) \\ &= \frac{2}{\sigma_b^2} \sum_{k=1}^{K} \frac{\sum_{m=1}^{M} Im\left(y_k \overline{Q_m} e^{-i\theta}\right) W_m\left(y_k, L^k, \theta\right)}{\sum_{m=1}^{M} W_m\left(y_k, L^k, \theta\right)} \end{aligned} \tag{5.33}$$

L'estimateur du maximum de vraisemblance serait théoriquement obtenu en annulant l'expression (5.33). Cependant, une recherche exhaustive de la solution de ce problème est irréalisable. C'est pourquoi on utilisera un procédé appelé "boucle d'estimation adaptative".

5.4.1.4 Boucle de phase

Dans le but d'approximer la solution du maximum de vraisemblance, on construit ce que l'on appelle une boucle de poursuite adaptative, qui corrige la valeur de θ à chaque nouveau symbole reçu. Cette boucle amène une simplification et un allègement considérable au niveau de l'implantation de l'algorithme. On déduit de (5.33) la boucle de phase, qui s'exprime simplement de la manière suivante :

$$\varphi_k = \varphi_{k-1} + \gamma \frac{\sum_{m=1}^{M} Im\left(y_k \overline{Q_m} e^{-i\varphi_{k-1}}\right) W_m\left(y_k, L^k, \varphi_{k-1}\right)}{\sum_{m=1}^{M} W_m\left(y_k, L^k, \varphi_{k-1}\right)} \tag{5.34}$$

Cette boucle est utilisable pour n'importe quelle modulation de type QAM, et exploite les informations a priori sur les symboles pouvant être fournies par exemple par un turbo décodeur, et qui sont désignées par le vecteur des log-vraisemblance L^k associé au symbole y_k. On remarquera que le terme W_m peut être vu comme un poids assigné à chaque symbole possible Q_m appartenant à la constellation QAM considérée. Ce poids dépend à la fois de la distance de Q_m vis-à-vis de l'observation et des log-vraisemblances des bits du symbole reçu (fournies par exemple par un turbo décodeur). Ainsi, une forte log-vraisemblance, ou une distance très faible par rapport au symbole reçu augmenteront le poids de ce symbole dans le calcul. La boucle tendra donc à calculer un nouveau θ qui permet de se rapprocher de ce symbole Q_m de poids important. A contrario, une faible log-vraisemblance associée à une grande distance par rapport à l'observation diminueront

l'importance d'un symbole QAM dans le calcul.

Si on considère le cas particulier $a_k = \{-1, +1\}$ on retrouve la boucle à remodulation souple pour une BPSK [Bro04]:

$$\varphi_k = \varphi_{k-1} + \gamma \frac{Im\left(y_k e^{-i\varphi_{k-1}}\right)\left[-\exp\left(-\frac{1}{2}L^k - \frac{2Re\left(y_k e^{-i\varphi_{k-1}}\right)}{\sigma_b^2} - \frac{1}{\sigma_b^2}\right) + \exp\left(\frac{1}{2}L^k + \frac{2Re\left(y_k e^{-i\varphi_{k-1}}\right)}{\sigma_b^2} - \frac{1}{\sigma_b^2}\right)\right]}{\exp\left(-\frac{1}{2}L^k - \frac{2Re\left(y_k e^{-i\varphi_{k-1}}\right)}{\sigma_b^2} - \frac{1}{\sigma_b^2}\right) + \exp\left(\frac{1}{2}L^k + \frac{2Re\left(y_k e^{-i\varphi_{k-1}}\right)}{\sigma_b^2} - \frac{1}{\sigma_b^2}\right)} \tag{5.35}$$

donc :

$$\varphi_k = \varphi_{k-1} + \gamma Im\left(y_k e^{-i\varphi_{k-1}}\right) \tanh\left(\frac{1}{2}L^k + \frac{2Re\left(y_k e^{-i\varphi_{k-1}}\right)}{\sigma_b^2}\right) \tag{5.36}$$

On peut donc utiliser le dispositif de la figure 5.23 pour réaliser l'estimation de phase. Lors de la première itération, l'estimation de la phase est effectuée selon (5.34) en considérant que l'on ne possède pas d'information a priori. Dans ce cas, LLR=0 au niveau de l'estimateur de la figure 5.23. Le bloc d'information grossièrement asservi en phase est alors appliqué à la suite de la chaîne de réception, c'est-à-dire le demappeur suivi du turbo décodeur. Dès la deuxième itération, il est possible de tenir compte dans l'estimation de l'information apportée par le turbo décodeur qui fournit des valeurs de log-vraisemblance associées à chacun des bits. Plusieurs itérations peuvent être nécessaires pour permettre la bonne convergence de l'algorithme.

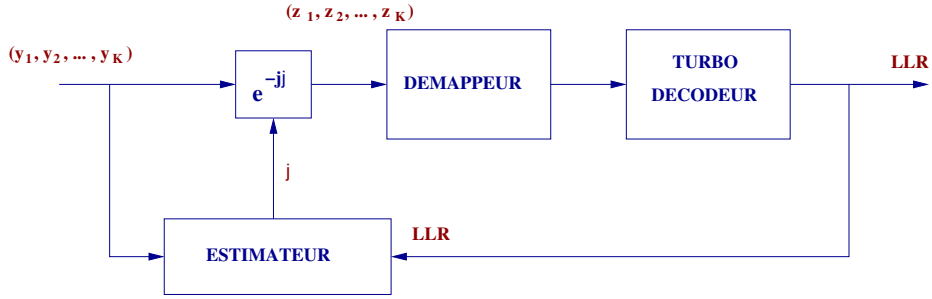


Figure 5.23: Boucle de phase à remodulation souple

5.4.1.5 Approximation de la boucle

Cas des faibles RSB Dans le cas de faibles RSB, on peut considérer que l'information souple délivrée par le turbo décodeur est peu fiable, c'est-à-dire que la valeur absolue des fiabilité est très faible. On peut alors prendre $L^k = 0$ dans la boucle (5.34).

Cas des forts RSB Lorsque le RSB est fort, les valeurs absolues des log-vraisemblances obtenues par turbo décodage tendent vers des valeurs élevées.

Par conséquent, le terme $\frac{1}{2}\sum_{n=1}^{N} q_n^m L_n^k$ devient prédominant quand on considère le symbole décidé $\hat{a}_k$ par demapping dur. Le terme concernant ce symbole devient prédominant dans la somme au numérateur et au dénominateur de l'équation (5.34) . Les autres termes devenant négligeables, on obtient finalement :

$$\varphi_k = \varphi_{k-1} + \gamma Im\left(y_k \overline{\hat{a}_k} e^{-i\varphi_{k-1}}\right) \tag{5.37}$$

Ceci constitue la boucle à remodulation dure "classique", c'est-à-dire qu'elle n'exploite que les décisions dures a_k sur les symboles et ne tient pas compte des informations souples fournies par le turbo décodeur.

5.4.1.6 Filtres correcteur et Boucle aller-retour

L'estimation des paramètres par de tels algorithmes de poursuite, dans le cas où ces paramètres ne sont pas constants, introduit généralement un biais. L'algorithme de poursuite ne peut pas rattraper parfaitement la vraie valeur du paramètre quand celui-ci évolue, ce qui amène une erreur de "trainage". Ce biais est généralement réduit par l'utilisation d'un pas d'adaptation variable. Cela permet de calculer un facteur γ en fonction du modèle d'évolution temporelle que l'on souhaite corriger. Pour des modèles simples on pourra se contenter d'un correcteur proportionnel simple, alors que dans des cas plus complexes, on pourra avantageusement recourir à un correcteur intégral, voire à un filtre d'ordre supérieur. De préference, le filtre γ pourra être réalisé au moyen d'un filtre numérique d'ordre 2 suivant la formule ci-aprés :

$$\gamma(z) = \gamma_1 + \frac{\gamma_2}{1 - z^{-1}} \tag{5.38}$$

et que l'on initialisera en tenant compte de tout les facteurs de continuité exploitables. Cependant, l'inconvénient d'un tel filtre est qu'il ralentit le système. Il est donc difficile de suivre des évolutions de phase rapides, comme un saut de phase par exemple. Ainsi, une seconde solution consiste à utiliser ce que nous appellerons une boucle "aller-retour" [Bro04] (cf. figure 5.24). Cette solution permet de bénéficier de l'effet de traitement par bloc en tenant compte des échantillons futurs.

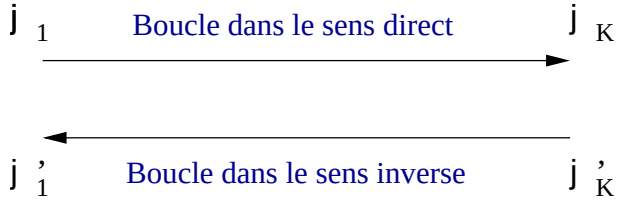


Figure 5.24: Boucle aller-retour pour une estimation de phase.

En effet, étant donné que l'on travaille de manière itérative sur un mot de code complet, et qu'il est donc nécessaire de conserver tous les symboles reçus en mémoire, on peut imaginer faire fonctionner la boucle dans les deux sens. En sens inverse, le biais apporté par la boucle de poursuite sera généralement opposé à celui obtenu dans

le sens direct. Ce phénomène est illustré par la figure 5.25. Ainsi, en faisant la moyenne pondérée de ces deux estimations pour chaque symbole, on peut espérer une réduction non négligeable en terme de biais, et donc en terme d'erreur quadratique moyenne. La boucle "retour" sera initialisée à la dernière valeur obtenu par la boucle "aller". La figure 5.24 illustre cette idée. Une première estimation de phase est donnée en faisant tourner la boucle dans le sens direct (ordre naturel des symboles transmis). On obtient donc un premier vecteur de valeurs de phase $\varphi_0, \ldots, \varphi_K$. Ensuite, on fait tourner la boucle dans le sens inverse pour obtenir un nouveau vecteur de valeurs de phase $\varphi'_0, \ldots, \varphi'_K$. On choisira $\varphi'_K = \varphi_K$, de manière à initialiser correctement la boucle de retour. De ce fait, l'estimation finale φ''_k pour chaque symbole sera donnée pour une transmission à blocs continus par :

$$\varphi''_k = \frac{\varphi_k + \varphi'_k}{2} \tag{5.39}$$

On peut voir sur la figure 5.26 l'effet de la boucle aller-retour sur l'estimation d'une phase pour une modulation 2-QAM (ou BPSK). La perturbation de phase est fixée pour cette simulation à $\sigma_\phi = \pi.10^{-2}$. On peut voir sur ces courbes que le gain en terme d'EQM approche les 3 dB pour $E_b/N_0 = 0$dB.

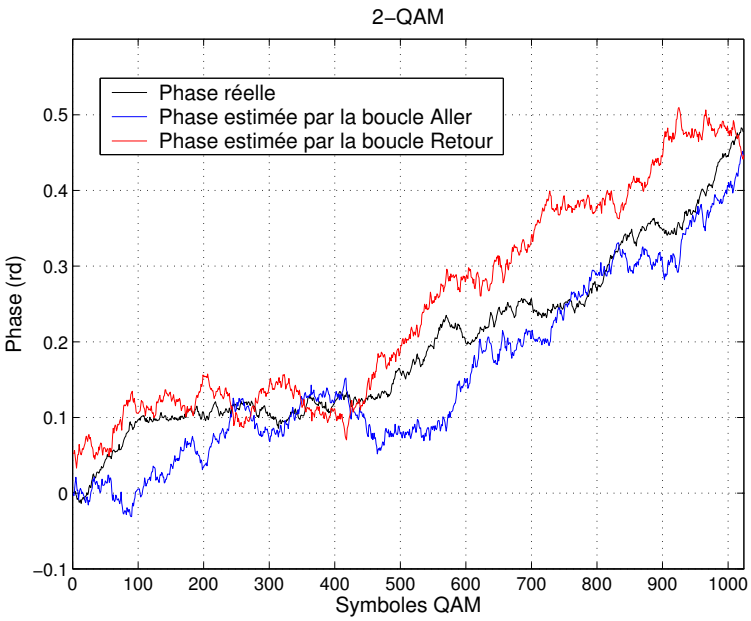


Figure 5.25: Boucles aller et retour sur une 2-QAM.

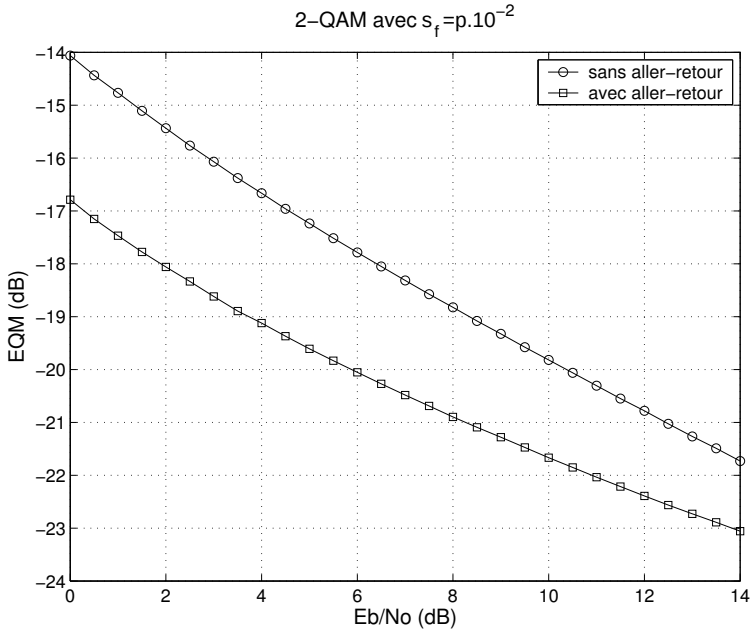


Figure 5.26: EQM pour une boucle aller-retour sur 2-QAM.

5.4.2 Boucle de gain

En règle générale, pour une modulation de type QAM, il faut également estimer le gain du canal afin de demapper correctement le symbole QAM. On suppose que le symbole reçu est de la forme :

$$y_k = Ga_k + b_k \tag{5.40}$$

pour $k = 1...K$ avec a_k le $k^{ème}$ symbole QAM émis et G le gain réel introduit par le canal. On suppose ici qu'il n'y a pas de problème de déphasage par rapport à l'émetteur. Alors, en considérant les observations $Y = (y_1, \cdots, y_K)^T$ on peut écrire la vraisemblance des observations en fonction du gain G :

$$P\left(Y \mid G, a_1, \cdots, a_K\right) = \prod_{k=1}^{K} \frac{1}{\pi\sigma_b^2} \exp\left(-\frac{|y_k - Ga_k|^2}{\sigma_b^2}\right) \tag{5.41}$$

On peut intégrer sur les symboles inconnus de la manière suivante :

$$P(Y \mid G) = \sum_{\substack{a_k \in \{Q_1, \cdots, Q_M\} \\ k = 1 \cdots K}} P\left(Y \mid G, a_1, \cdots, a_K\right) \times P\left(a_1, \cdots, a_K\right) \tag{5.42}$$

En considérant l'équation (5.29) on obtient :

$$P\left(Y \mid G\right) = \left(\frac{1}{\pi\sigma_b^2}\right)^K \prod_{k=1}^{K} \left[\left(\prod_{n=1}^{N} \frac{1}{2\cosh\left(\frac{L_n^k}{2}\right)} \right) \exp\left(-\frac{|y_k|^2}{\sigma_b^2}\right) \sum_{m=1}^{M} \exp\left(\frac{1}{2}\sum_{n=1}^{N} q_n^m L_n^k + \frac{2Re\left(y_k \overline{Q_m} G\right) - G^2|Q_m|^2}{\sigma_b^2} \right) \right] \tag{5.43}$$

Pour alléger les notations, on définit le poids d'un point de la constellation QAM de manière analogue à celui utilisé pour la boucle de phase :

$$W_m\left(y_k, L^k, G\right) = exp\left(\frac{1}{2}\sum_{n=1}^{N} q_n^m L_n^k - \frac{\left|y_k - GQ_m\right|^2}{\sigma_b^2} \right) \tag{5.44}$$

L'équation (5.43) se met alors sous la forme :

$$P\left(Y \mid G\right) = \left(\frac{1}{\pi\sigma_b^2}\right)^K \prod_{k=1}^{K} \left[\left(\prod_{n=1}^{N} \frac{1}{2\cosh\left(\frac{L_n^k}{2}\right)} \right) \sum_{m=1}^{M} W_m\left(y_k, L^k, G\right) \right] \tag{5.45}$$

On peut alors deriver le logarithme de cette expression par rapport à G:

$$\frac{\partial}{\partial G}\log P\left(Y \mid G\right) = \frac{2}{\sigma_b^2}\sum_{k=1}^{K} \frac{\sum_{m=1}^{M}\left(Re\left(y_k\overline{Q_m}\right) - G\left|Q_m\right|^2\right) W_m\left(y_k, L^k, G\right)}{\sum_{m=1}^{M} W_m\left(y_k, L^k, G\right)} \tag{5.46}$$

On obtient finalement le maximum de vraisemblance en annulant (5.46) et on en déduit la boucle de gain suivante :

$$G_k = G_{k-1} + \gamma \frac{\sum_{m=1}^{M}\left(Re\left(y_k\overline{Q_m}\right) - G_{k-1}\left|Q_m\right|^2\right) W_m\left(y_k, L^k, G_{k-1}\right)}{\sum_{m=1}^{M} W_m\left(y_k, L^k, G_{k-1}\right)} \tag{5.47}$$

qui permet d'estimer récursivement le gain du canal de transmission pour chacun des symboles.

5.4.3 Boucle de Gain-Phase

L'inconvenient des deux boucles précédentes est que chacune d'elles suppose que l'autre paramètre est connu et compensé. Or dans la réalité, chacune d'elle se baserait sur le résultat de l'autre boucle. Si l'un des paramètres n'est pas correctement estimé, on risque de propager des erreurs. Il est donc préférable d'estimer ces deux paramètres scalaires de manière conjointe. En effet, en considérant les deux perturbations, on peut mettre le symbole reçu sous la forme :

$$y_k = a_k G e^{i\theta_k} + b_k \quad \text{pour } k = 1 \ldots k \tag{5.48}$$

La vraisemblance des observations en fonction de ces deux paramètres inconnus est donnée par :

$$P\left(Y \mid \theta, G, a_1, \cdots, a_K\right) = \prod_{k=1}^{K} \frac{1}{\pi\sigma_b^2} \exp\left(-\frac{\left|y_k - Ge^{-i\theta}a_k\right|^2}{\sigma_b^2}\right) \tag{5.49}$$

Ceci nous permet, en intégrant sur les symboles inconnus selon la même démarche que pour les deux boucles précédentes, d'obtenir l'expression suivante :

$$P\left(Y \mid \theta, G\right) = \left(\frac{1}{\pi\sigma_b^2}\right)^K \prod_{k=1}^{K} \left[\left(\prod_{n=1}^{N} \frac{1}{2\cosh\left(\frac{L_n^k}{2}\right)}\right) \sum_{m=1}^{M} W_m\left(y_k, L^k, \theta, G\right)\right] \tag{5.50}$$

où le poids $W_m\left(y_k, L^k, \theta, G\right)$ associé au point Q_m est défini par :

$$W_m\left(y_k, L^k, \theta, G\right) = exp\left(\frac{1}{2}\sum_{n=1}^{N} q_n^m L_n^k - \frac{\left|y_k - Ge^{i\theta}Q_m\right|^2}{\sigma_b^2}\right) \tag{5.51}$$

On calcule la dérivée de (5.50) par rapport aux deux paramètres afin d'obtenir le maximum de vraisemblance :

$$\frac{\partial}{\partial\theta}\log P\left(Y \mid \theta, G\right) = \frac{2}{\sigma_b^2}\sum_{k=1}^{K} \frac{\sum_{m=1}^{M} Im\left(y_k\overline{Q_m}Ge^{-i\theta}\right) W_m\left(y_k, L^k, \theta, G\right)}{\sum_{m=1}^{M} W_m\left(y_k, L^k, \theta, G\right)} \tag{5.52}$$

$$\frac{\partial}{\partial G}\log P\left(Y \mid \theta, G\right) = \frac{2}{\sigma_b^2}\sum_{k=1}^{K} \frac{\sum_{m=1}^{M} \left(Re\left(y_k\overline{Q_m}e^{-i\theta}\right) - G\left|Q_m\right|^2\right) W_m\left(y_k, L^k, \theta, G\right)}{\sum_{m=1}^{M} W_m\left(y_k, L^k, \theta, G\right)} \tag{5.53}$$

Finalement on obtient les boucles de poursuite conjointe sur la phase et le gain :

$$\varphi_k = \varphi_{k-1} + \gamma\frac{\sum_{m=1}^{M} Im\left(y_k\overline{Q_m}G_{k-1}e^{-i\varphi_{k-1}}\right) W_m\left(y_k, L^k, \varphi_{k-1}, G_{k-1}\right)}{\sum_{m=1}^{M} W_m\left(y_k, L^k, \varphi_{k-1}, G_{k-1}\right)} \tag{5.54}$$

$$G_k = G_{k-1} + \gamma\frac{\sum_{m=1}^{M} \left(Re\left(y_k\overline{Q_m}e^{-i\varphi_{k-1}}\right) - G_{k-1}\left|Q_m\right|^2\right) W_m\left(y_k, L^k, \varphi_{k-1}, G_{k-1}\right)}{\sum_{m=1}^{M} W_m\left(y_k, L^k, \varphi_{k-1}, G_{k-1}\right)} \tag{5.55}$$

La figure 5.27 représente le dispositif permettant de mettre en oeuvre cet estimateur. Comme pour le dispositif permettant l'estimation de la phase seule, celui-ci considère dans un premier temps qu'il ne possède aucune information a priori sur les symboles. Il estime uniquement à l'aide des symboles reçus et de manière conjointe le gain et la phase pour chaque symbole grâce à (5.54) et (5.55) où $L^k = 0$. Les symboles, corrigés par les paramètres de phase et de gain estimés de façon grossière, sont alors demappés et décodés. Ensuite le turbo décodeur fournit une information a priori sur les symboles à l'estimateur. Il tire alors parti de cette information pour affiner son estimation.

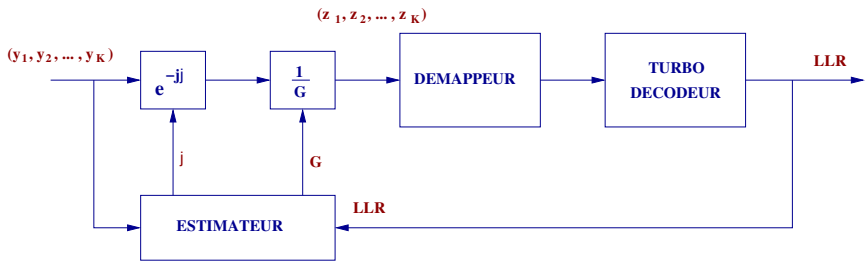


Figure 5.27: Boucle de gain-phase à remodulation souple

5.4.4 Résultats de simulation

Nous allons ici nous intéresser aux performances en terme d'EQM des boucles à remodulation souple. Un turbo décodeur BCH(32,26,4)^2 fournira les log-vraisemblances nécessaires à la deuxième itération. Afin d'évaluer les performances, nous comparerons les résultats à une boucle connaissant les symboles émis, qui donne théoriquement les meilleurs performances possibles. Il est important dans notre cas de correctement estimer l'importance du "dosage" de l'information souple dans la boucle de retour. En effet, si les valeurs absolues des fiabilités sont très faibles, elles ne pèseront que très peu dans le poids W_m des symboles et la remodulation sera équivalente à la première boucle sans information. A contrario, si les valeurs absolues sont très importantes, le point de la QAM correspondant aux signes des LLR fournis par le décodeur aura un poids très prédominant dans le calcul. La boucle sera alors équivalente à une remodulation dure utilisant des décisions dures sur la sortie du turbo décodeur.

5.4.4.1 Estimation de phase

Il est donc nécessaire de trouver un compromis sur la quantité d'information réinjectée dans l'itération suivante et c'est pourquoi les poids seront redéfinis selon :

$$W_m\left(y_k, L^k, \theta\right) = exp\left(K_{DFL}\sum_{n=1}^{N} q_n^m L_n^k - \frac{\left|y_k - e^{i\theta}Q_m\right|^2}{\sigma_b^2}\right) \tag{5.56}$$

où K_{DFL} est le coefficient de pondération des LLR qui doit être optimisé en fonction des conditions de transmissions.

Pour nos simulations, nous modélisons la perturbation sur la phase par une gigue gaussienne, comme décrit par l'équation (5.6). La variance de la gigue est notée σ_ϕ^2 et nous n'introduisons pas de dérive linéaire.

Dans un premier temps, nous étudions l'influence de ce coefficient dans le cas d'une estimation de phase sur une modulation 2-QAM. On peut s'apercevoir sur la figure 5.28 que le coefficient K_{DFL} doit prendre une valeur suffisament élevée pour obtenir le minimum d'EQM. Cela signifie que dans ce cas une décision dure est suffisante. On peut voir sur la figure 5.29 le résultat en terme d'EQM en fonction de E_b/N_0. Dans ce cas, la décision souple ne donnant pas de meilleurs résultats que la décision dure, seule la boucle à remodulation dure apparait sur la figure 5.29.

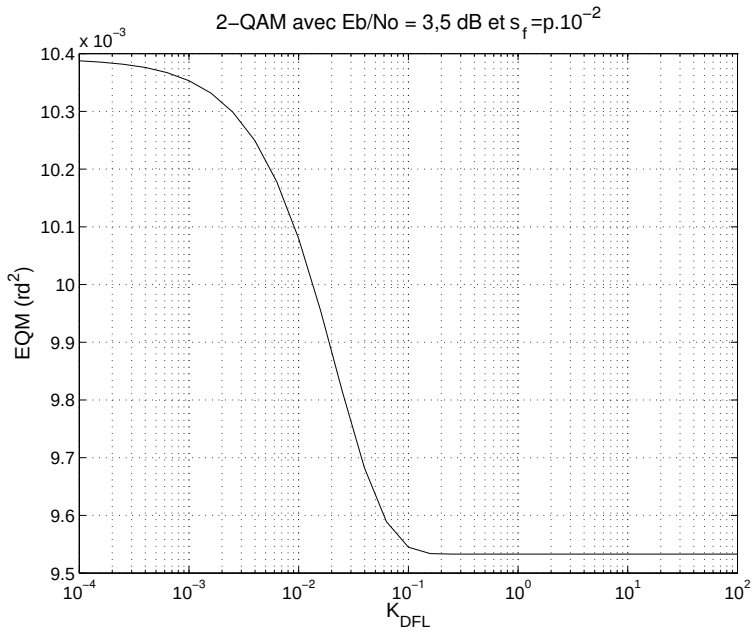


Figure 5.28: Erreur quadratique moyenne en fonction du coefficient de pondération des LLR.

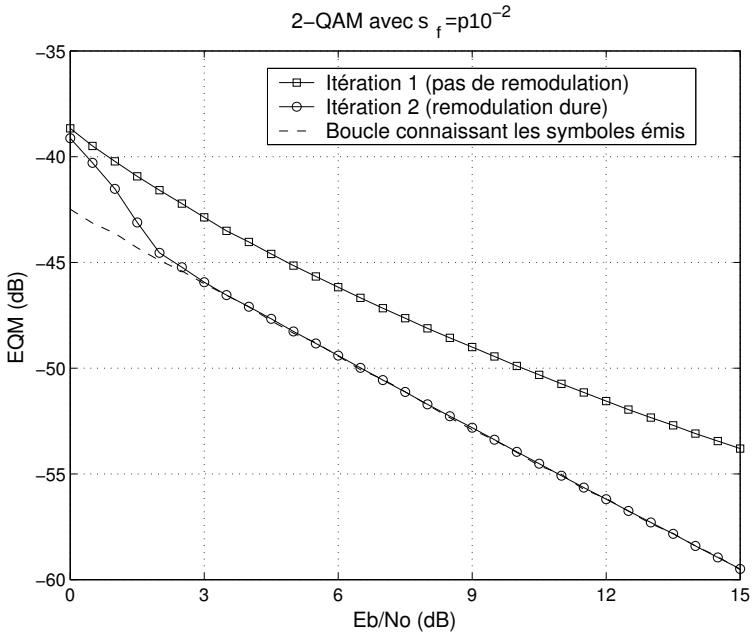


Figure 5.29: Erreur Quadratique Moyenne pour une 2-QAM.

Par contre, pour une modulation à grand nombre d'états, le résultat est différent. La figure 5.30 par exemple montre le cas d'une 1024-QAM à 19 dB ($TEB = 10^{-8}$ pour un

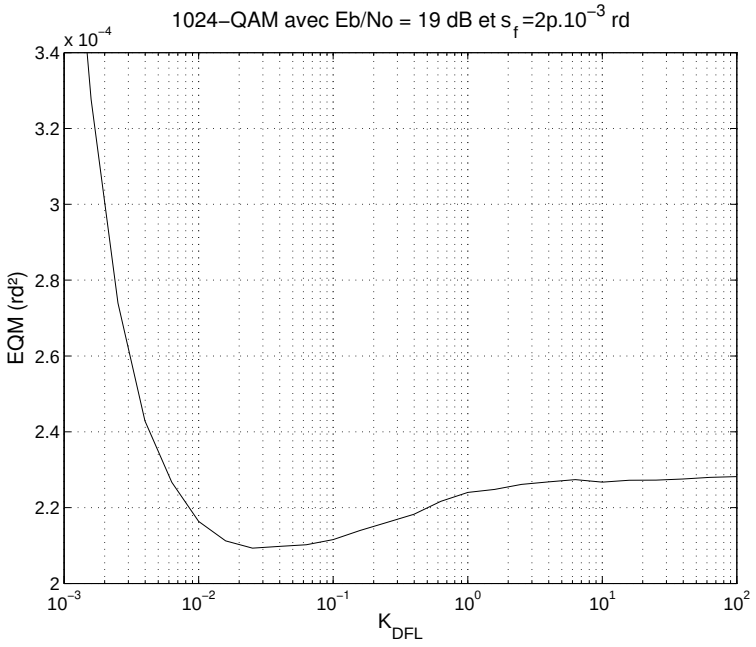


Figure 5.30: Erreur quadratique moyenne en fonction du coefficient de pondération des LLR.

code BCH(32,26,4)^2) avec une gigue de phase $\sigma_\phi = 2\pi.10^{-3}$rd. La figure 5.31 présente le cas d'une 256-QAM à $E_b/N_0 = 13$dB ($TEB = 3.10^{-5}$) avec une gigue de phase $\sigma_\phi = 6\pi.10^{-3}$rd.

L'intérêt de la décision souple apparaît clairement sur les courbes 5.30 et 5.31. En effet, on peut voir que dans le cas ou K_{DFL} est très faible, la valeur de l'EQM augmente, car on est dans le cas d'une boucle ne disposant pas d'informations a priori. Lorsque K_{DFL} tend vers une grande valeur, on tend vers le résultat d'une boucle à remodulation dure utilisant une décision dure sur la sortie du turbo décodeur. Par contre, autour de $K_{DFL} = 3.10^{-2}$pour la figure 5.30 et autour de $K_{DFL} = 2.10^{-1}$ pour la figure 5.31 on observe un minimum qui correspond au meilleur "dosage" possible de la quantité d'information souple réinjectée.

Dans le cas de la 256-QAM, on peut voir sur la figure 5.32 l'apport de la décision souple. Dans un premier temps, on voit que la deuxième itération, qui utilise l'information provenant du turbo décodeur, apporte un gain en terme d'EQM de 4dB quand $E_b/N_0 = 13$dB par rapport à une boucle sans connaissance à priori. Ensuite, le fait d'utiliser l'information souple permet de gagner 0.4 dB par rapport à une boucle n'utilisant que les décisions dures sur les bits. La troisième itération, tirant partie de l'estimation des LLR de l'itération précédente, permet d'affiner encore l'estimation et donc de diminuer l'EQM. L'amélioration apportée par une remodulation souple par rapport à la remodulation dure est de 0.6 dB. On peut supposer que des itérations supplémentaires apporteront une amélioration de l'estimation et un écart supérieur entre boucle à remodulation dure et souple.

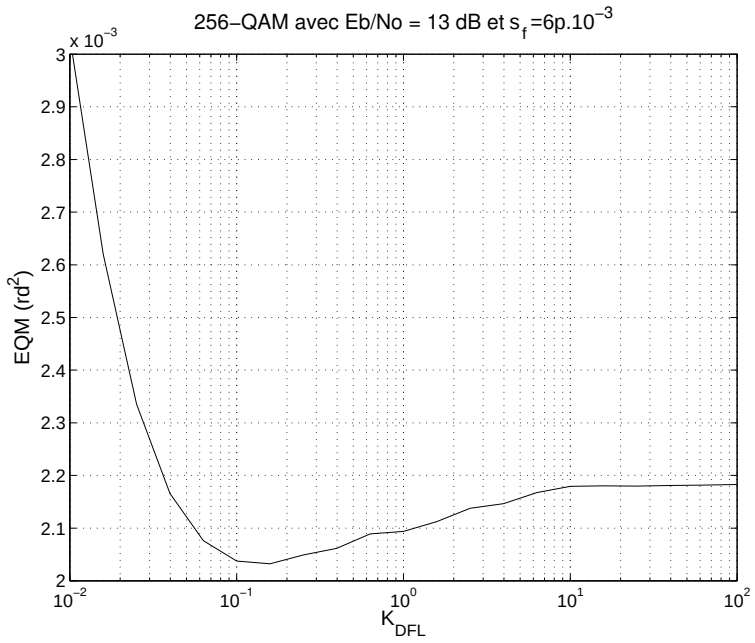


Figure 5.31: Erreur quadratique moyenne en fonction du coefficient de pondération des LLR.

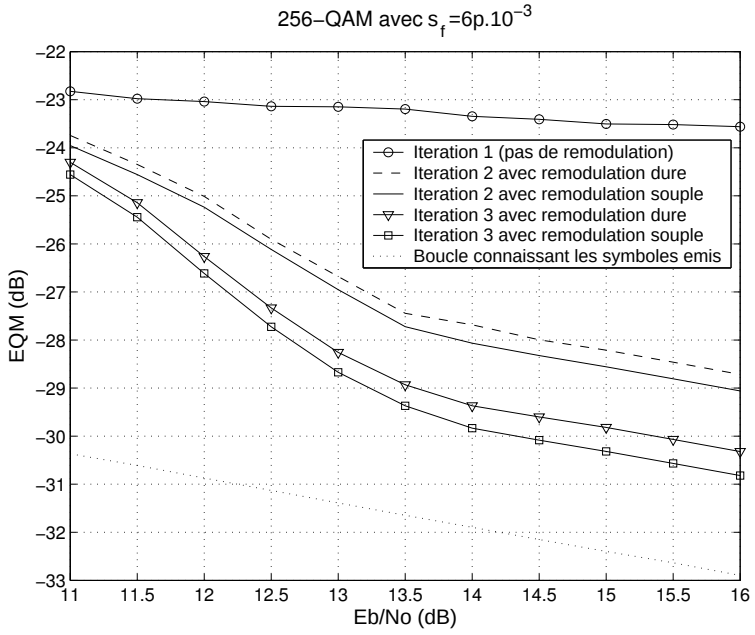


Figure 5.32: Erreur Quadratique Moyenne pour une 256-QAM.

5.4.4.2 Estimation conjointe du gain et de la phase

Dans ce cas, si on introduit le coefficient de pondération K_{DFL}, les poids sont alors définis par :

$$W_m\left(y_k, L^k, G\right) = exp\left(K_{DFL}\sum_{n=1}^{N} q_n^m L_n^k - \frac{|y_k - GQ_m|^2}{\sigma_b^2}\right) \tag{5.57}$$

Afin d'estimer les performances de l'algorithme d'estimation conjointe de phase et gain, on modélise l'évolution de la phase comme dans le paragraphe précédent par une gigue gaussienne. L'évolution du gain est modélisée comme pour la phase par un mouvement brownien, et le gain G_k appliqué par le canal au symbole y_k est donné par cette expression :

$$G_k = G_{k-1} + g_k \tag{5.58}$$

où g_k est une variable gaussienne de variance σ_G^2. Cette modélisation du gain ne correspondant pas à un canal réel, il serait intéressant d'effectuer le même type de simulations sur un canal de type Rayleigh par exemple.

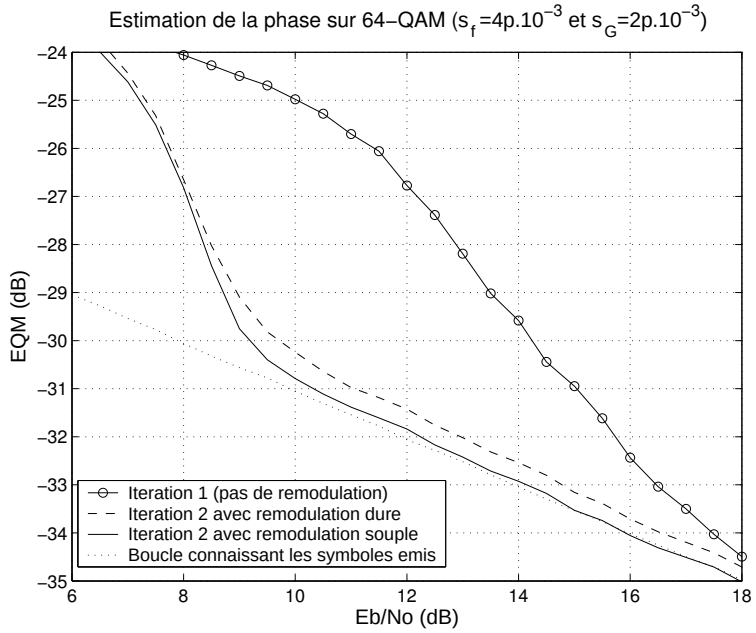


Figure 5.33: Erreur quadratique moyenne sur l'estimation de la phase.

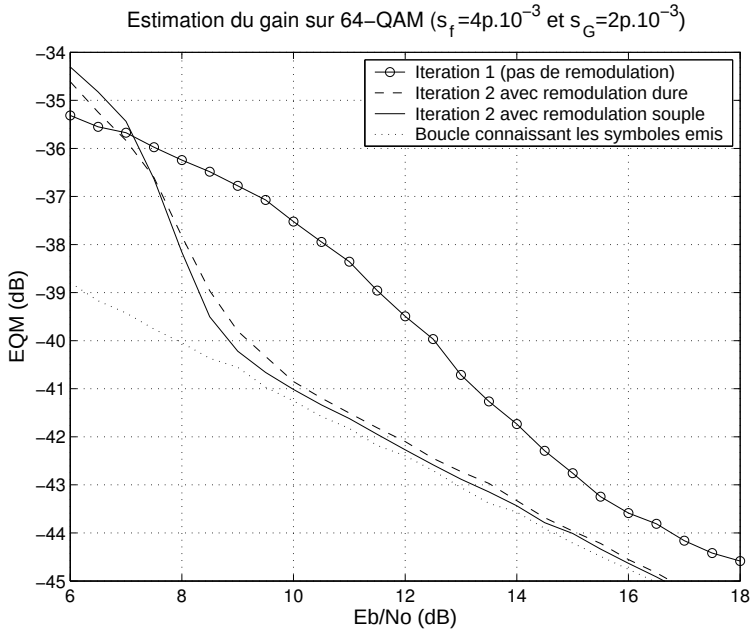


Figure 5.34: Erreur quadratique moyenne sur l'estimation du gain.

Nous étudions ici le cas d'une modulation 64-QAM. La gigue de phase a dans ce cas pour écart-type $\sigma_\phi = 4\pi.10^{-3}$ et la perturbation sur le gain a pour écart-type $\sigma_G = 2\pi.10^{-3}$. Les figures 5.33 et 5.34 montrent l'erreur quadratique moyenne obtenue sur la phase, en fonction de E_b/N_0, pour plusieurs configurations de l'algorithme.

On peut tout d'abord voir l'évolution de l'EQM pour le cas où on ne réalise que la première itération, sans information a priori. Dans ce cas, l'algorithme est beaucoup moins performant que la boucle idéale, c'est à dire la boucle qui connait les symboles émis.

Aprés la première itération, on peut tirer parti des informations données par le turbo décodeur. Un premier cas est obtenu en utilisant la décision dure sur les log-vraisemblances. Dans ce cas, on se rapproche de la boucle idéale, mais pour l'estimation de la phase en particulier il reste un écart constant d'environ 0,5 dB.

Par contre, dans le cas de la remodulation souple, qui exploite toute l'information disponible à la sortie du turbo décodeur, la boucle tend vers la boucle idéale. On obtient donc un gain sur l'EQM de la phase de 0,5 dB dans la partie droite de la courbe, et le gain est de 0.7 dB pour $E_b/N_0 = 9$ dB ($TEB = 10^{-7}$).

La figure 5.35 illustre le résultat de la transmission en terme de taux d'erreur binaire. Bien que les courbes d'erreurs quadratiques moyennes (figures 5.33 et 5.34) donnent des performances quasiment optimales dès la deuxième itération, il faut ici un nombre d'itérations supérieur pour s'approcher du cas idéal. Ce phénomène peut être expliqué par le fait que, bien que l'EQM soit faible, elle ne donne aucune information sur la forme de la distribution de l'erreur d'estimation. Il est donc probable que cette distribution ne soit pas gaussienne et possède des queues de distribution épaisses. Ainsi, des erreurs relativement importantes sur quelques symboles, mais suffisament rares pour n'avoir

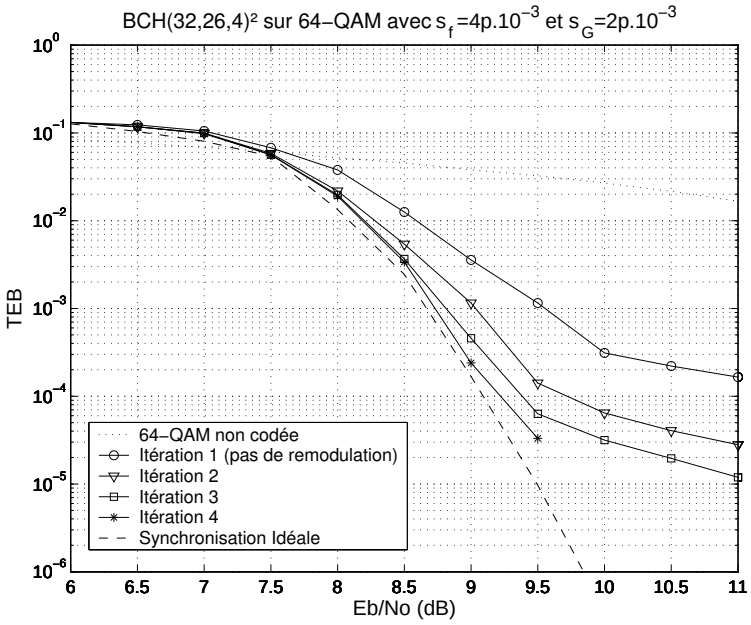


Figure 5.35: Taux d'erreur binaire pour une estimation conjointe de la phase et du gain.

aucune influence sur l'EQM, suffisent pour dégrader le taux d'erreur binaire. Quelques itérations supplémentaires permettent alors de se rapprocher du cas idéal en affinant l'estimation sur ces symboles en particulier.

5.5 Conclusion

Nous avons présenté ici deux systèmes de synchronisation innovants. Le premier exploite de manière globale l'information disponible sur la totalité d'un mot de code afin d'estimer la phase des symboles de ce mot. Ce système de turbo synchronisation exploite une propriété des turbo codes particulièrement intéressante. Il permet d'obtenir des performances comparables à d'autres algorithmes aveugles sur des constellations moyennes, et apporte d'excellents résultats sur les constellations de grande tailles. Il permet en effet de s'approcher du BER obtenu dans la cas d'une synchronisation parfaite dans des conditions difficiles où d'autres algorithmes ne permettent que des performances passables. Il est robuste aux dérives et fonctionne à des rapports signal sur bruit aussi bas que les turbo codes le permettent. Il ne présente de plus pas l'ambiguité de $\pi/2$ qui apparait dans la plupart des systèmes de synchronisation. Il fonctionne en aveugle, sans séquence d'appentissage, mais en ayant un a priori sur les données jusqu'à des RSB proches de la limite Shannon grâce à l'effet turbo. Il est surtout très simple à mettre en oeuvre vu qu'il exploite directement les valeurs de sortie d'un turbo décodeur dont la conception peut être rendue très légère grâce à la structure simplifiée. Ainsi, il se trouve être un concept tout à fait intéressant dans le cadre d'une application haut-débit nécessitant l'utilisation de grandes constellations. Le deuxième algorithme présenté a pour avantage

d'estimer une phase pour chaque symbole reçu. Ainsi, il permet de poursuivre des évolutions de phase rapides à l'intérieur d'un mot de code. L'estimation est grandement améliorée par l'utilisation des informations provenant du turbo décodeur. En particulier, pour l'estimation conjointe de deux paramètres, cette boucle permet d'atteindre des performances très proches de la boucle idéale ayant la connaissance des symboles. Ces avantages en font un algorithme novateur très intéressant pour les systèmes haut débits utilisant de grandes constellations. Les deux algorithmes présentés ont d'ailleurs donné lieu à la publication de deux brevets [GBBV04c][GBBV04b].

Chapitre 6

Conclusions et perspectives

Le travail rapporté dans ce document a concerné l'utilisation des turbo codes blocs dans les communications haut débit et l'estimation de certains paramètres de transmission. Nous avons ainsi developpé des algorithmes qui ont donné lieu à des brevets et des publications.

Aprés avoir exposé le principe des turbo codes blocs, nous avons dans un premier temps présenté un algorithme permettant une simplification importante de la mise en oeuvre du décodeur de R.Pyndiah. Cette solution permet en particulier une économie importante au niveau de la quantité de mémoire nécessaire à la réalisation d'un tel dispositif. Ainsi, l'intégration d'un turbo décodeur dans un système réel serait facilitée par l'utilisation de ce procédé.

Ensuite, nous avons démontré par simulation l'intérêt de l'utilisation des turbo codes dans un système de communication numérique particulier, le VDSL. En effet, l'utilisation de ce schéma de codage apporte des gains non négligeables en terme de débit et de distance, ce qui permettrait d'améliorer le service fournis à l'utilisateur. Le système serait ainsi capable de fournir un profil de débit donné à un plus grand nombre d'abonnés, permettant ainsi au fournisseur d'accés de proposer des offres intéressantes au plus grand nombre. Dans le contexte actuel où la concurrence règne entre les opérateurs de télécommunication, tout avantage en terme de couverture est intéressant.

Nous avons également présenté un algorithme de synchronisation qui possède la particularité d'exploiter l'information contenue dans la moyenne des fiabilités en sortie d'un turbo décodeur. Il présente en outre des avantages sur la plupart des algorithmes classiques, comme le fait qu'il ne présente pas d'ambiguité de phase, ni de glissement de cycle. Il ne nécessite pas de séquence d'apprentissage car il exploite la redondance contenue dans les mots de code. Particulièrement adapté aux systèmes nécessitant des tailles de constellations importantes, il est parfaitement adapté aux systèmes de communications haut débit.

Enfin, le deuxième algorithme de synchronisation présenté ici est une boucle qui utilise une remodulation souple, c'est-à-dire une boucle qui exploite l'information de fiabilité pour chacun de bits fournis par le turbo décodeur. Par le fait qu'elle estime une nouvelle phase pour chacun des symboles QAM reçus, cette boucle est capable de poursuivre des évolutions de phase particulièrement rapides. Il est possible d'estimer conjointement le

gain et la phase du signal reçu, ce qui permet d'obtenir de très bonnes performances. Un tel système peut se réveler trés intéressant pour des communications sur des canaux difficiles dont les paramètres évoluent rapidement.

Diverses perspectives se dégagent pour compléter et poursuivre ce travail :

- Il est possible d'associer les deux algorithmes de synchronisation dans le but de profiter des avantages de chacun. En effet, le premier algorithme permettrait de détecter et corriger les décrochements éventuels du second.

- Il serait intéressant de développer une méthode simple permettant d'estimer la valeur optimale du coefficient de pondération des LLR dans la boucle à remodulation souple. Il faudrait pouvoir estimer ce paramètre de manière systématique en fonction des informations disponibles lors d'une transmission afin d'optimiser l'intérêt de cette boucle.

- Il serait également intéressant de disposer d'une optimisation du pas de correction, par exemple en introduisant un filtre de Kalman.

- Il serait souhaitable de tester l'algorithme à remodulation souple phase et gain dans des conditions de fading de type rayleigh.

- D'autres boucles à remodulation souple ont été imaginées et calculées dans le cadre de ce travail, notamment une boucle d'estimation aveugle des paramètres du canal, et une boucle d'estimation de l'instant optimal d'échantillonage. Ces deux boucles exploitent strictement le même principe que la boucle de phase à remodulation souple. Il est d'ailleurs possible de réaliser une boucle estimant conjointement phase, instant d'échantillonnage et réponse du canal. Une étude approfondie de ces boucles est nécessaire. Il est probable que l'augmentation du nombre de paramètres à estimer rendrait plus intéressant encore l'apport de la remodulation souple.

Glossaire

ADSL : Asymmetric Digital Subscriber Line
AWGN : Additionnal White Gaussian Noise
BBGA : Bruit Blanc Gaussien Additif
BCH : Bose, Chaudhuri, Hocquenghem
BER : Bit Error Rate
BPSK : Binary Phase Shift Keying
CAN : Convertisseur Analogique Numérique
CNA : Convertisseur Numerique Analogique
CO : Central Office
CRB : Cramer Rao Bound
DFL : Decision Feedback Loop
DMT : Discrete Multi Tone
EQM : Erreur Quadratique Moyenne
FFT : Fast Fourier Transform
FTTB : Fiber To The Building
FTTCab : Fiber To The Cabinet
FTTEx : Fiber To The Exchange
FTTH : Fiber To The Home
IFFT : Inverse Fast Fourier Transform
IIS : Inter Symbole Interference
ISI : Interference Inter Symbole
LLR : Log Likelihood Ratio
LT : Line Termination
MAP : Maximum A Posteriori
MAQ : Modulation d'Amplitude en Quadrature
MCRB : Modified Cramer Rao Bound
MSE : Mean Square Error
MV : Maximum de Vraisemblance
NT : Network Termination
OFDM : Orthogonal Frequency Division Multiplexing
ONU : Optical Network Unit
PSK : Phase Shift Keying
QAM : Quadrature Amplitude Modulation
RS : Reed Solomon

RSB : Rapport Signal à Bruit
SDFL : Soft Decision Feedback Loop
SISO : Soft In Soft Out
SNR : Signal to Noise Ratio
TCM : Treillis Coded Modulation
TEB : Taux d'Erreur Binaire
TFD : Transformée de Fourier Discrète
TFDI : Transformée de Fourier Discrète Inverse
VDSL : Very high bit rate Digital Subscriber Line

Bibliographie

[ANS98] *Network and customer installation interfaces, Assymetric Digital Subscriber Line (ADSL), Metallic Interface*, 1998.

[AP00] P. Adde and R. Pyndiah. Recent simplifications and improvements in Block Turbo Codes. In *2nd International Symposium on Turbo Codes and related Topics*, pages 133–136, September 2000.

[BAG02] J.M. Brossier, P.O. Amblard, and B. Geller. Self adaptative PLL for multi-carrier local loop transmission. In *Proc. Eusipco*, pages 631–634, Toulouse, September 2002.

[Bat98] Battail. *Introduction à la théorie de l'information.* Masson, 1998.

[Ber82] Berlekamp. *Algebraic coding theory.* Aegan Press, Laguna Hills, 1982.

[BG96] C. Berrou and A. Glavieux. Near optimum error correcting coding and decoding : Turbo-codes. *IEEE Trans. Commun.*, 44(10) :1261–1271, October 1996.

[BGT93] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error correcting and decoding turbo codes. In *IEEE ICC'93*, pages 1004–1007, May 1993.

[Bin90] J.A.C. Bingham. Multi-carrier modulation for data transmission : an idea whose time has come. *IEEE Communication Magazine*, 28(5) :5–14, May 1990.

[Bin00] J.A.C. Bingham. *ADSL, VDSL and Multicarrier modulation.* Wiley-Interscience, 2000.

[BRC60] R.C. Bose and D. Ray-Chaudhuri. On a class of error correcting binary codes. *Inform. Control*, 3 :68–79, March 1960.

[Bro04] J.M. Brossier. Procédé d'estimation de la phase dans un système de communication numérique et boucle à verrouillage de phase, Avril 2004.

[CDG92] Cohen, Dornstetter, and Godlewski. *Codes correcteur d'erreurs.* Masson, 1992.

[Cha72] D. Chase. A class of algorithms for decoding block codes with channel measurement information. *IEEE Trans. Inform. Theory*, 18(1) :170–182, January 1972.

[Che95] W.Y. Chen. Broadcast Digital Subscriber Lines. *IEEE Journal on Selected Areas in Communications*, 13(9), December 1995.

[CMBL00] J. Cambonie, P. Mejean, D. Barthela, and J. Lienard. Procédé et dispositif de calcul d'une transformée rapide de Fourier pour une modulation OFDM, Decembre 2000.

[Com96] J.P. Combes. *Micro-Ondes*. Dunod, 1996.

[CT91] Cover and Thomas. *Elements of information theory*. Wiley, New-York, 1991.

[DMR94] A.N. D'Andrea, U. Mengali, and R. Reggiannini. The modified Cramér-Rao bounds and its applications to synchronization problems. *IEEE Trans. Commun.*, 44 :1391–1399, Feb./Mar./Apr. 1994.

[Dup89] J. Dupraz. *Théorie du signal et transmission de l'information*. Eyrolles, 1989.

[eLP94] Eric Walter et Luc Pronzato. *Identification de modèles paramétriques*. MASSON, 1994.

[ETS03a] *Transmission and multiplexing access transmission systems on metallic access cables ; Very high bit rate Digital Subscriber Line (VDSL), Part 1 : functionnal requirement*, 2003.

[ETS03b] *Transmission and multiplexing access transmission systems on metallic access cables ; Very high bit rate Digital Subscriber Line (VDSL), Part 2 : transceiver specifications*, 2003.

[For65] Forney. On decoding BCH codes. *IEEE Transactions on Information Theory*, IT-11 :549–557, October 1965.

[For66] G.D. Jr Forney. *Concatenated Codes*. Cambridge MA, MIT Press, 1966.

[GBBV04a] B. Geller, J.P. Barbot, J.M. Brossier, and C. Vanstraceele. Procédé de décodage itératif de codes blocs et dispositif décodeur correspondant, Juin 2004.

[GBBV04b] B. Geller, J.P. Barbot, J.M. Brossier, and C. Vanstraceele. Procédé d'estimation de la phase de données d'observation transmises sur un canal de transmission en modulation QAM, Septembre 2004.

[GBBV04c] B. Geller, J.P. Barbot, J.M. Brossier, and C. Vanstraceele. Système de compensation de phase pour turbo décodeur, Juin 2004.

[Gel01] B. Geller. *Codage canal et codes correcteurs*. polycopié de cours DEA SIPT et ENSERG, 2001.

[GRM98] Fluvio Gini, Rugerro Reggiannini, and Umberto Mengali. The modified Cramér-Rao bound in vector parameter estimation. *IEEE Trans. Commun.*, 46(1) :52–60, January 1998.

[HHM01] S.A. Hirst, B. Honary, and G. Markarian. Fast Chase algorithm with an application in turbo decoding. *IEEE Trans. Commun.*, 49(10) :1693–1699, October 2001.

[Hoc59] A. Hocquenghem. Codes correcteurs d'erreurs. *Chiffres*, 2 :147–156, 1959.

[HOP96] S. Hagenauer, E. Offer, and L. Papke. Iterative decoding of binary block and convolutional codes. *IEEE Trans. Inform. Theory*, 42(2) :429–445, March 1996.

[HWK00] L. Hanzo, W.T. Webb, and T. Keller. *Single- and Multi-carrier Quadrature Amplitude Modulation : Principles and Applications for Personal Communications, WLANs and Broadcasting.* Wiley, 2000.

[Iss] 0. Isson. *Traitement numérique du signal en modulation multiporteuse appliqué au modem VDSL.* thèse INPG soutenue le 4 octobre 2002 à Grenoble.

[Kal89] I. Kalet. The multitone channel. *IEEE Transactions on Communication*, 37(2) :119–124, February 1989.

[LC83] Lin and Costello. *Error control coding : fundamentals and applications.* Prentice Hall, Englewood Cliffs, 1983.

[LM04] V. Lottici and M.Luise. Embedding carrier phase recovery into iterative decoding of turbo-coded linear modulations. *IEEE Transactions on Communications*, 52 :661 – 669, April 2004.

[LSWZ96] V.B. Lawrence, L.J. Smithwik, J.J. Werner, and N.A. Zervos. Broadband access to the home of copper. *Bell Labs Technical Journal*, 1 :100–114, July 1996.

[LYHH93] J. Lodge, R. Young, P. Hoeher, and J. Hagenauer. Separable MAP 'filters' for the decoding of product and concatenated codes. In *IEEE ICC'93*, pages 1740–1745, May 1993.

[Mas69] Massey. Shift register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, 15 :122–137, 1969.

[MdJ94] M. Moeneclaey and G de Jonghe. ML oriented NDA carrier synchronization for general rotationally symmetric signal constellations. *IEEE Trans. Commun.*, 42 :2531–2533, August 1994.

[Moe98] M. Moeneclaey. On the true and the modified Cramér-Rao bounds for the estimation of a scalar parameter in the presence of nuisance parameters. *IEEE Trans. Commun.*, 46 :1536–1544, November 1998.

[NSM03] N. Noels, H. Steendam, and M. Moeneclaey. The Cramer-Rao Bound for Phase Estimation for Coded Linearly Modulated Signal. *IEEE Commun. Letters*, 7(5) :207–209, May 2003.

[OC01] Wrangok Oh and Kyungwhoon Cheun. Joint Decoding and Carrier Phase Recovery Algorithm for Turbo Codes. *IEEE Commun. Letters*, 5(9) :375–377, September 2001.

[PGPJ94] Pyndiah, Glavieux, Picart, and Jacq. Near optimum decoding of product codes. In *IEEE Globecom 94*, pages 339–343, San Fransisco, December 1994.

[PW72] Peterson and Weldon. *Error correcting codes.* MIT Press, Cambridge, 1972.

[Pyn98] R. Pyndiah. Near optimum decoding of product codes : Block Turbo Codes. *IEEE Trans. Commun.*, 46(8) :1003–1010, August 1998.

[RS60] I. Reed and G. Solomon. Polynomial codes over certain fields. *J. Soc. App. Math.*, 8 :300–304, June 1960.

[SCS99] T. Starr, J.M. Cioffi, and P.J. Silverman. *Digital Subscriber Line Technology*. Prentice Hall, Upper Saddle River, NJ, 1999.

[Sha48] C.E. Shannon. A mathematical theory of communication. *Bell System Tech. J.*, 27 :379–423, July 1948.

[TLFL01] Tang, Liu, Fossorier, and Lin. On combining Chase 2 and GMD decoding algorithms for non binary block codes. *IEEE Communication letters*, 5(5) :209–211, May 2001.

[Ung82] G. Ungerboeck. Channel coding with multilevel phase signals. *IEEE Trans. Inform. Theory*, IT 28(1) :55–67, Jan 1982.

[VBA+03] C. Vanstraceele, J.P. Barbot, L.N. Atallah, B. Geller, and J.M. Brossier. Estimation de l'erreur de phase d'une modulation QAM conjointement au décodage itératif d'un code bloc. In *Proc. GRETSI'2003 (19 th Symposium)*, volume 2, pages 363–366, Paris, September 2003.

[VBBG03] C. Vanstraceele, J.M. Brossier, J.P. Barbot, and B. Geller. A New Iterative Phase Tracking Scheme. In *Proc. ICICS'2003*, volume On CDroom 1C4.7, pages 1–4, Singapore, December 2003.

[VBGB02] C. Vanstraceele, J.P. Barbot, B. Geller, and J.M. Brossier. BCH turbo codes for xDSL transmission. In *Proc.IEEE of InOWo'02*, volume 1, pages 263–266, Hamburg, September 2002.

[VGBB02] C. Vanstraceele, B. Geller, J.P. Barbot, and J.M. Brossier. Block Turbo Codes for Multicarrier Local Loop Transmission. In *Proc. VTC'2002 Fall*, volume 3, pages 1775–1778, Vancouver, September 2002.

[VGBB04a] C. Vanstraceele, B. Geller, J. P. Barbot, and J. M. Brossier. Joint estimation of QAM carrier phase with block turbo decoding. In *Proc. IEEE VTC'2004 Spring*, volume On CDrom, pages 1–5, Milan, Mai 2004.

[VGBB04b] C. Vanstraceele, B. Geller, J.P. Barbot, and J.M. Brossier. An iterative phase synchronization scheme for general QAM constellations. In *Proc. ICC'2004*, volume CT 08 Synchronization, pages 519–522, Paris, June 2004.

[Vit67] A.J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Info. Theory*, IT-13 :260–269, 1967.

[WE71] S.B. Weinstein and P.M. Ebert. Data transmission by frequency-division multiplexing using the discrete fourier transform. *IEEE Transactions on Communications Technology*, COM-19(5) :628–634, October 1971.

[Wei84a] L.F. Wei. Rotationally invariant convolutional channel coding with expended signal space. part i : 180°. *IEEE JSAC*, SAC-2 :659–671, September 1984.

[Wei84b] L.F. Wei. Rotationally invariant convolutional channel coding with expended signal space. part ii : Nonlinear codes. *IEEE JSAC*, SAC-2 :672–686, September 1984.